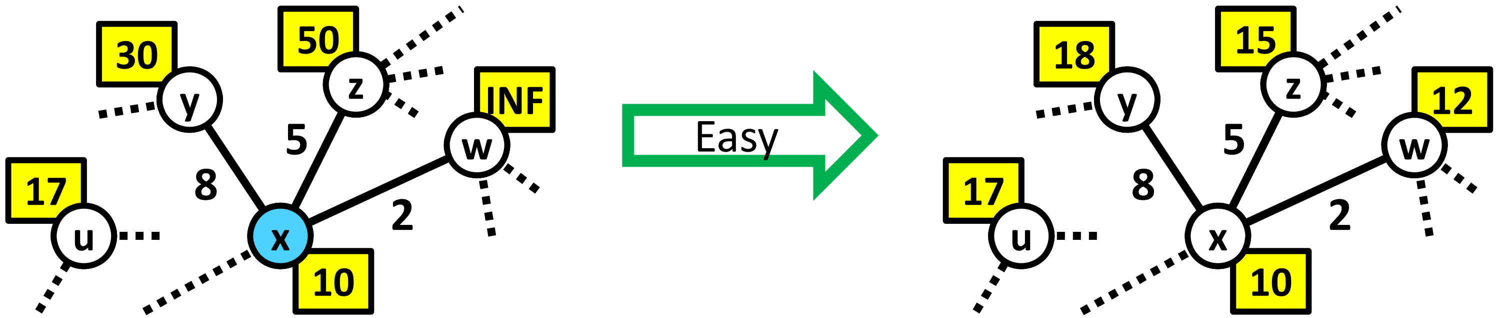
Notes on Minimum Spanning Tree problem

A Priority queue trouble

Usual theoretical and conceptual descriptions of Prim's (Dijkstra's, etc.) algorithm say:
"... store nodes in a priority queue".
Technically, this is nearly impossible to do.
The graph and the nodes are defined separately and stored elsewhere in the memory.
The node has no reference to its position in the queue.
The programmer does not know where is the node in the queue.
So, how to move a node inside the queue according to the algorithm demands?
Standard solution:
Do not move a node, enqueue a "copy of a node", possibly more times.

When a copy of the node with the smallest value (=highest priority) among all its copies appears at the top of the queue
it does its job exactly according to the algorithm prescription.
From that moment on, all other copies of the node which are still in the queue become useless and must be ignored. The easiest way to ignore a copy is to check it when it later appears at the top of the queue: If the node is already closed, ignore the copy, pop it and process the next top of the queue. If the node is still open, process it according to the algorithm.



top priority queue

x	..	...	u	...	y	...	z	...	w	...
10	..	...	17	...	30	...	50	...	INF	...

Difficult

top priority queue

...	..	w	...	z	...	u	...	y	...
...	..	12	...	15	...	17	...	18	...

q.insert(y);
q.insert(z);
q.insert(w);
// push the nodes
// once more to the queue.

Easy

top priority queue

..	w	...	z	...	u	...	y	...	y	...	z	...	w	...
..	12	...	15	...	17	...	18	...	30	...	50	...	INF	...

new copies old copies

The older copies of nodes will get to the top of the queue later than the new copies (which have higher priority) . The older copy gets to the top when the node had been processed and closed earlier. Thus:
If the node at the top of the queue is closed just pop it and do not process it any more.

Example of Prim's algorithm implementation using standard library priority queue

```
void MST_Prim(Graph g, int start, final int [] dist, int [] pred ) {
  // allocate structures
  int currnode = start, currdist, neigh;
  int INF = Integer.MAX_VALUE;
  boolean [] closed = new boolean[g.N];

  PriorityQueue <Integer> pq
    = new PriorityQueue<Integer>( g.N,
          new Comparator<Integer>() {
            @Override
            public int compare(Integer n1, Integer n2) {
              if( dist[n1] < dist[n2] ) return -1;
              if( dist[n1] > dist[n2] ) return 1;
              return 0;
          } }  );

  // initialize structures
  pq.add( start );
  for( int i = 0; i < g.N; i++ ) pred[i] = i;
  Arrays.fill( dist, INF );
  Arrays.fill( closed, false );
  dist[start] = 0;
```

Example of Prim's algorithm implementation using standart library priority queue

```
  for( int i = 0; i < g.N; i++ ) {
    // take the closest node and skip the closed ones
    while( closed[currnode = pq.poll()] == true );
    // and expand the closest node
    for( int j = 0; j < g.dg[currnode]; j++ ){
      neigh = g.edge[currnode][j];
      if( !closed[neigh] &&
         ( dist[neigh] > g.w[currnode][j]) ) {
          dist[neigh] = g.w[currnode][j];
          pred[neigh] = currnode;
          pq.add(neigh);
      }
    } // for j
    closed[currnode] = true;
  } // for i
} // MST_Prim
```

A very small change produces Dijkstra's algorithm:

```
    if( !closed[neigh] &&
       ( dist[neigh] > g.w[currnode][j] + dist[currnode]) ) {
         dist[neigh] = g.w[currnode][j] + dist[currnode];
```

Prim and Dijkstra Algorithms compared

The only difference: $\boxed{}$

```
void Dijkstra( Graph G, function weight, Node startnode )
  for each u in G.V:  u.dist = INFINITY; u.parent = NIL
  startnode.dist = 0; PriorityQueue Q = G.V

  while !Q.isEmpty()
    u = Extract-Min(Q)
    for each v in G.Adj[u]
      if (v in Q) and v.dist > weight(u,v) + u.dist
        v.parent = u
        v.dist = weight(u,v) + u.dist
```

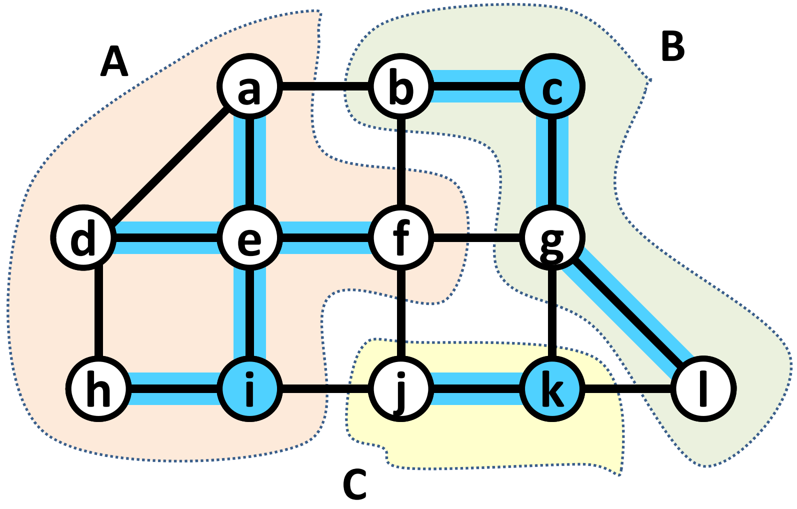
```
void MST_Prim( Graph G, function weight, Node startnode )
  for each u in G.V:  u.dist = INFINITY; u.parent = NIL
  startnode.dist = 0; PriorityQueue Q = G.V

  while !Q.isEmpty()
    u = Extract-Min(Q)
    for each v in G.Adj[u]
      if (v in Q) and v.dist > weight(u,v)
        v.parent = u
        v.dist = weight(u,v)
```

Disjoint-set data structure alias Union-Find structure

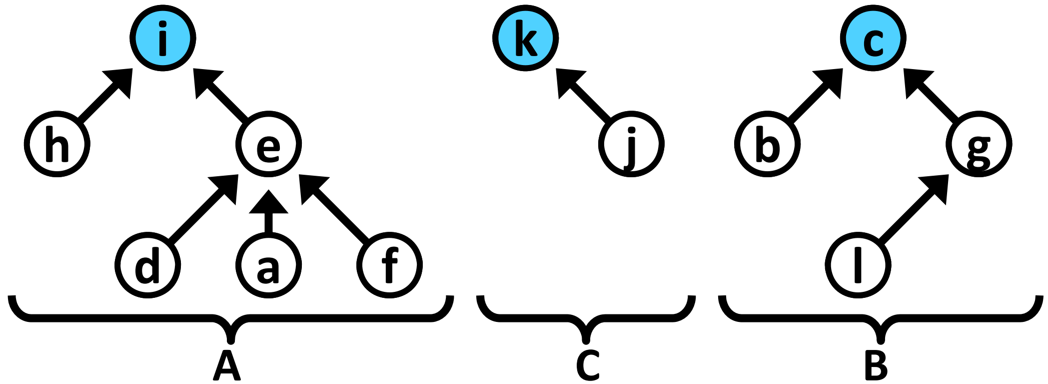
A graph with spanning trees
of its subgraphs A, B, C.

Each subgraph is represented by
one tree in the Union-Find structure.



Union-Find structure
with trees corrresponding
to subgraphs A, B, C.

Tree roots represent the subgraphs
and/or their spanning trees.



Implementation of the
Union-Find structure.

node	a	b	c	d	e	f	g	h	i	j	k	l
boss	e	c	c	e	i	e	c	i	i	k	k	g

A tree root is called
here "boss", for brevity.

Disjoint-set data structure alias Union-Find structure

Only 3 operations are needed:

Initialize()

Union(representativeA, representativeB) // merges the two sets represented // by the given two representatives

Find(nodeX) // returns a representative of the set to which X belongs

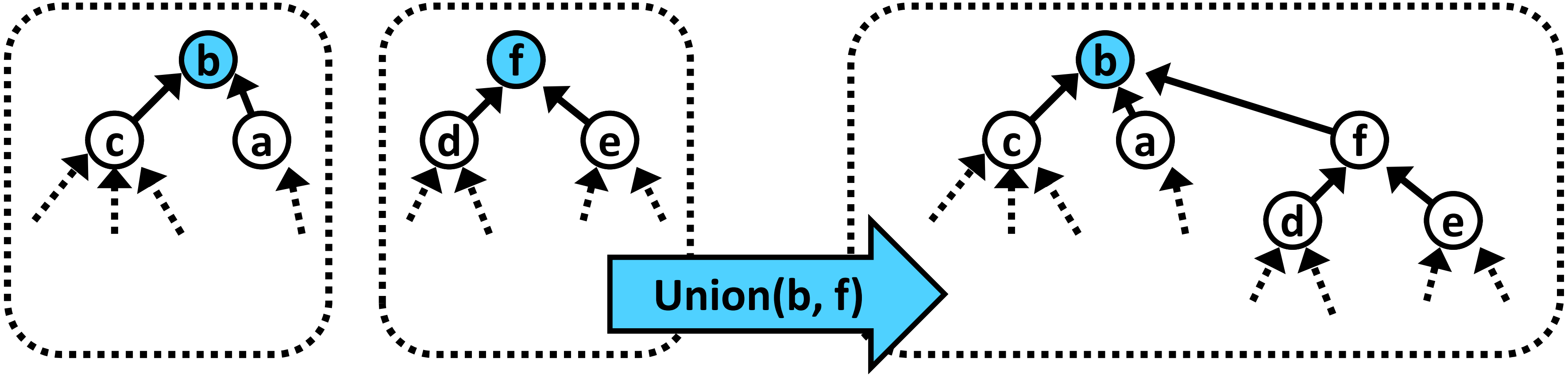
```
int [] boss;  // = representative = tree root
int [] rank;

void UF_init( int n ) {
  boss = new int [n];
  rank = new int [n];
  for( int i = 0; i < n; i++ ) {
    boss[i] = i;   // everybody's their own boss
    rank[i] = 0;   // initial rank is 0
} }
```

Easy experiment, try it at home:

When the end nodes of the inspected edges are chosen uniformly randomly then the average depth of a queried node in the Union-Find forest is less than 2.

Union with rank comparison

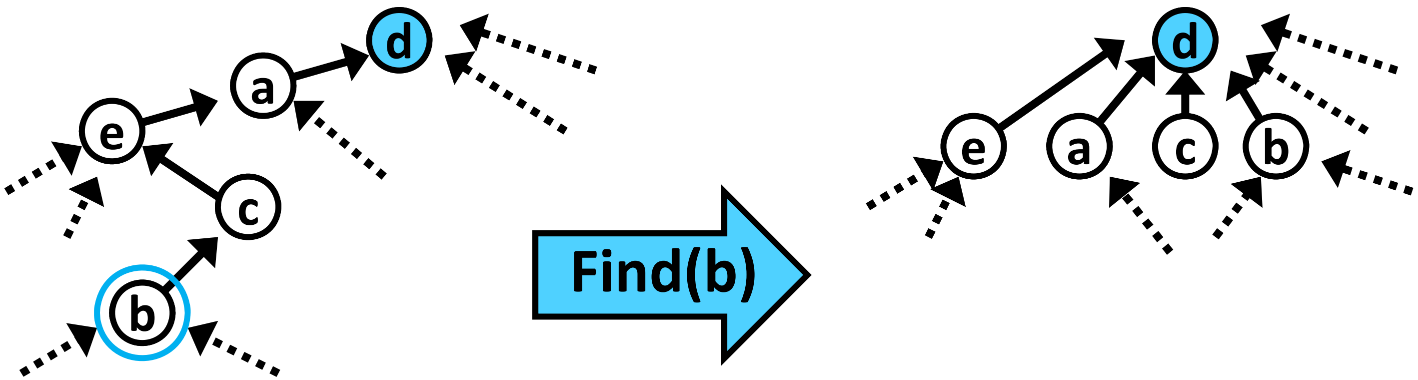


node	...	a	...	b	...	c	...	d	...	e	...	f	...
boss	...	b	...	b	...	b	...	f	...	f	...	f	...
rank	...		...		...		...		...		...		...

node	...	a	...	b	...	c	...	d	...	e	...	f	...
boss	...	b	...	b	...	b	...	f	...	f	...	b	...
rank	...		...	?	...		...		...		...		...

```
void UF_union( int rootA, int rootB ) {
   if( rank[rootB] > rank[rootA] )
         boss[rootA] = rootB;
   else {
      boss[rootB] = rootA;
      if( rank[rootB] == rank[rootA] )     // change rank?
         rank[rootA]++;
} }
```

Find with path compression



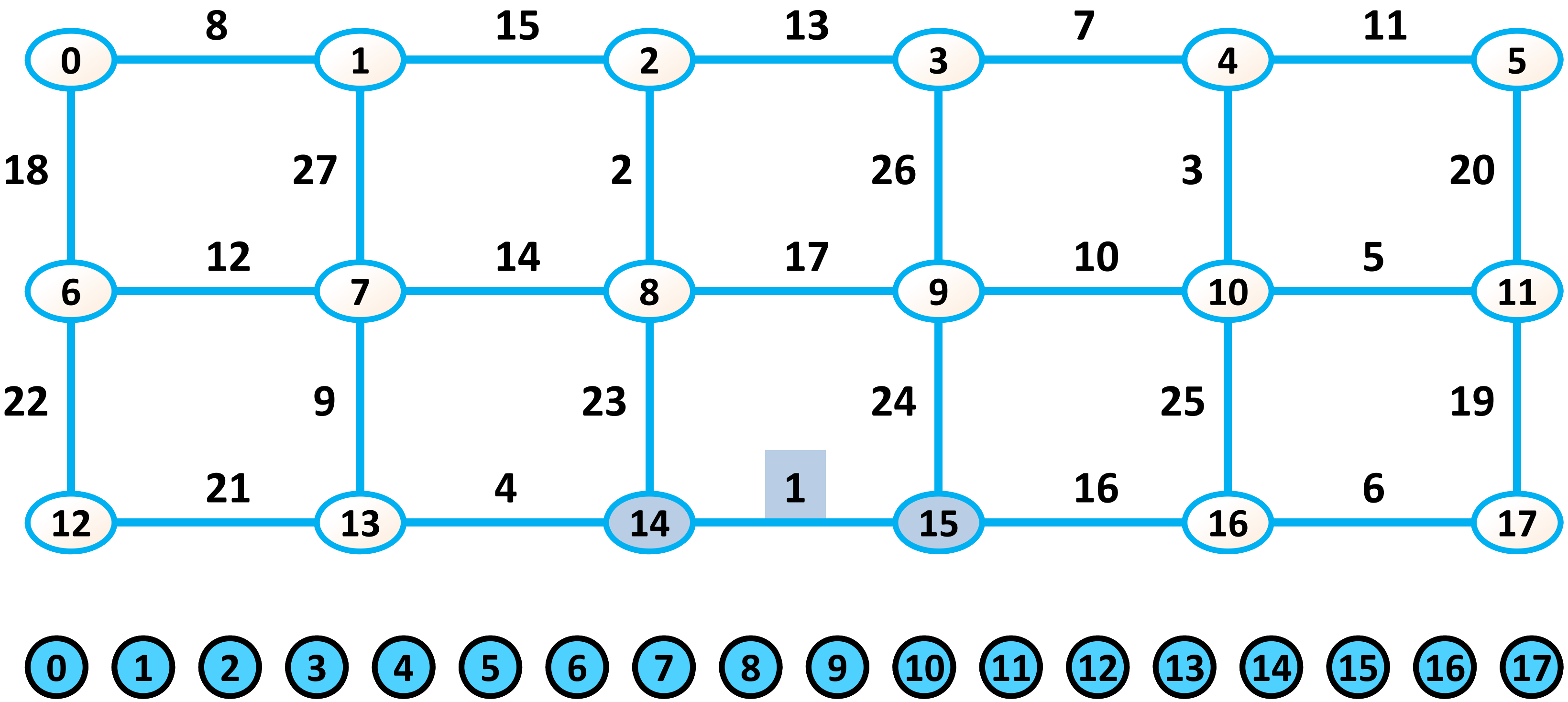
node	...	a	...	b	...	c	...	d	...	e	...
boss	...	d	...	c	...	e	...	d	...	a	...

node	...	a	...	b	...	c	...	d	...	e	...
boss	...	d	...	d	...	d	...	d	...	d	...

```c
int UF_find( int a ) {
  int parent = boss[a];
  if( parent != a )
    boss[a] = UF_find( parent ); // path compression
  return boss[a];
}
```

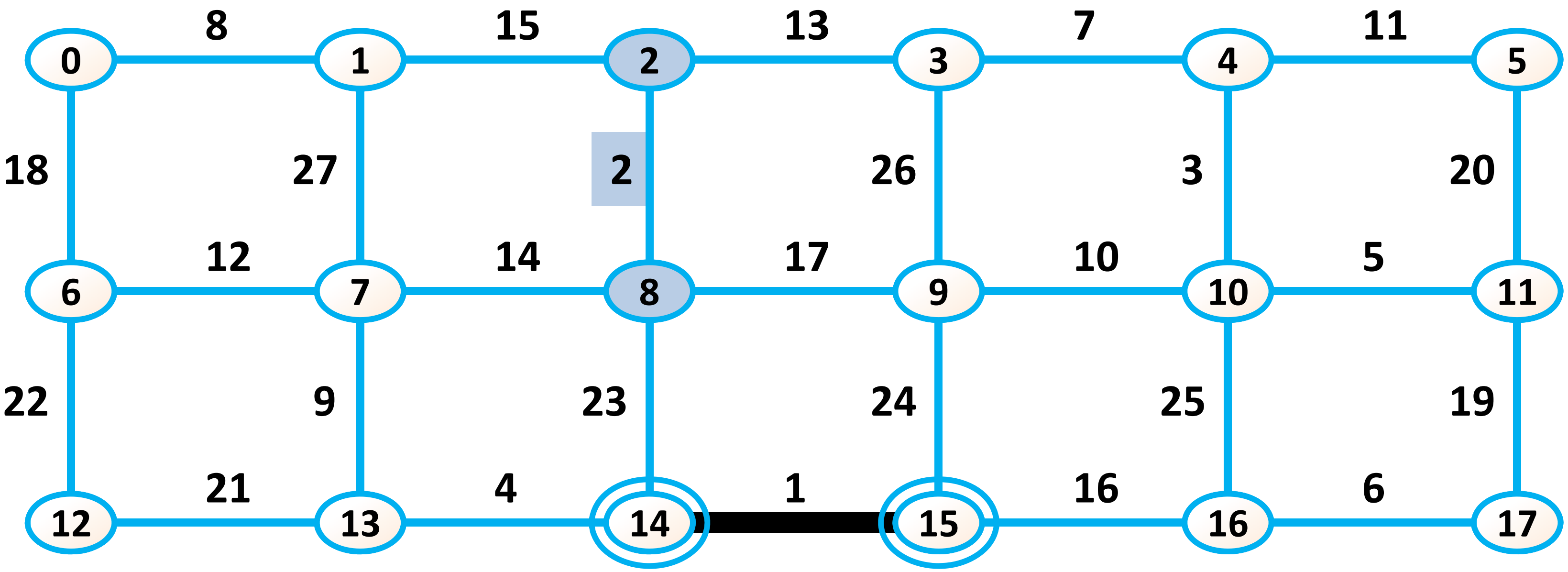
```c
int find( int a )  // for C experts
 { return( boss[a] == a ? a : (boss[a] = find(boss[a])) );}
```

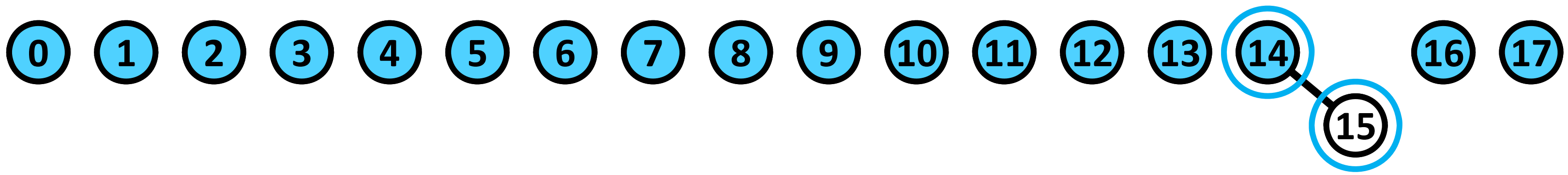
Kruskal's algorithm and Union-Find scheme example



node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
rank	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

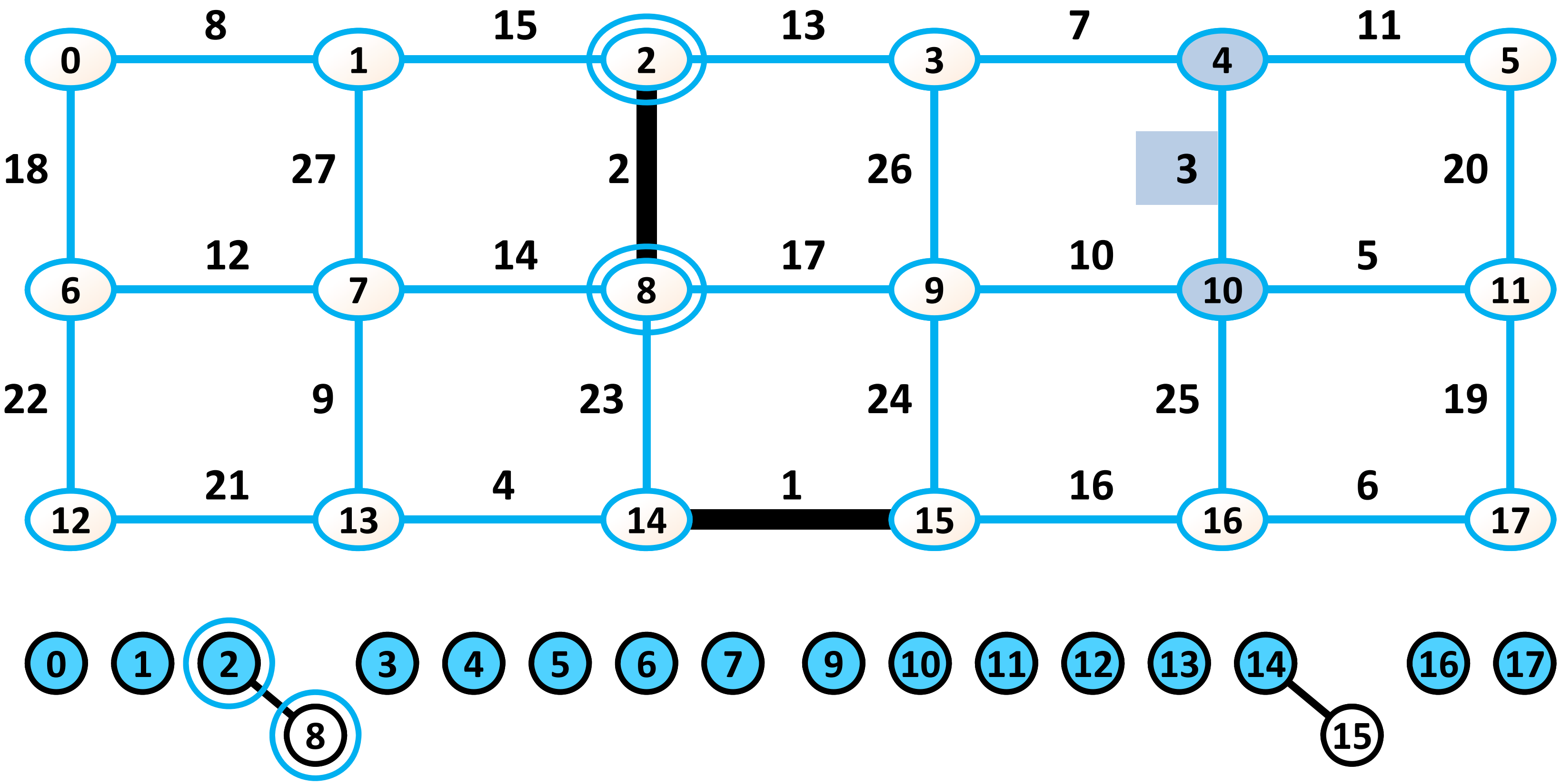
Kruskal's algorithm and Union-Find scheme example





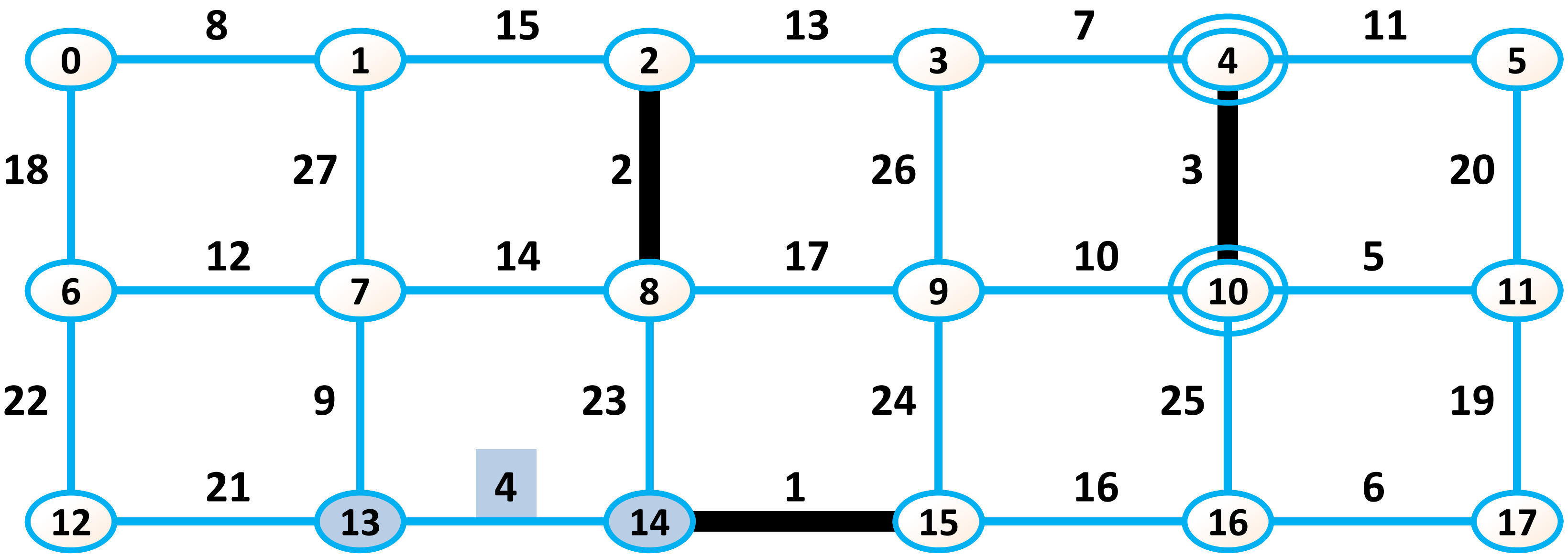
node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	14	16	17
rank	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

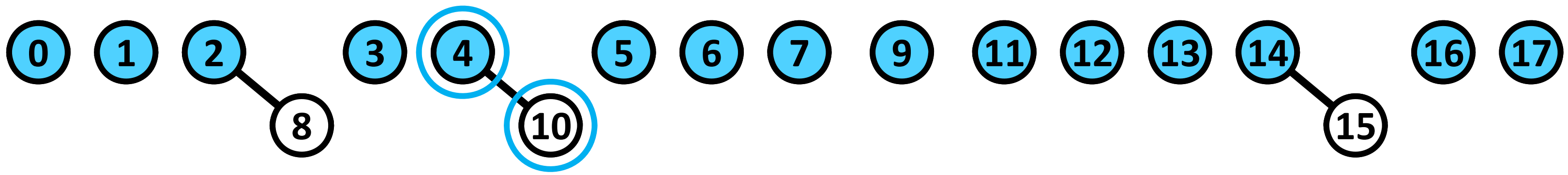
Kruskal's algorithm and Union-Find scheme example



node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	1	2	3	4	5	6	7	2	9	10	11	12	13	14	14	16	17
rank	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

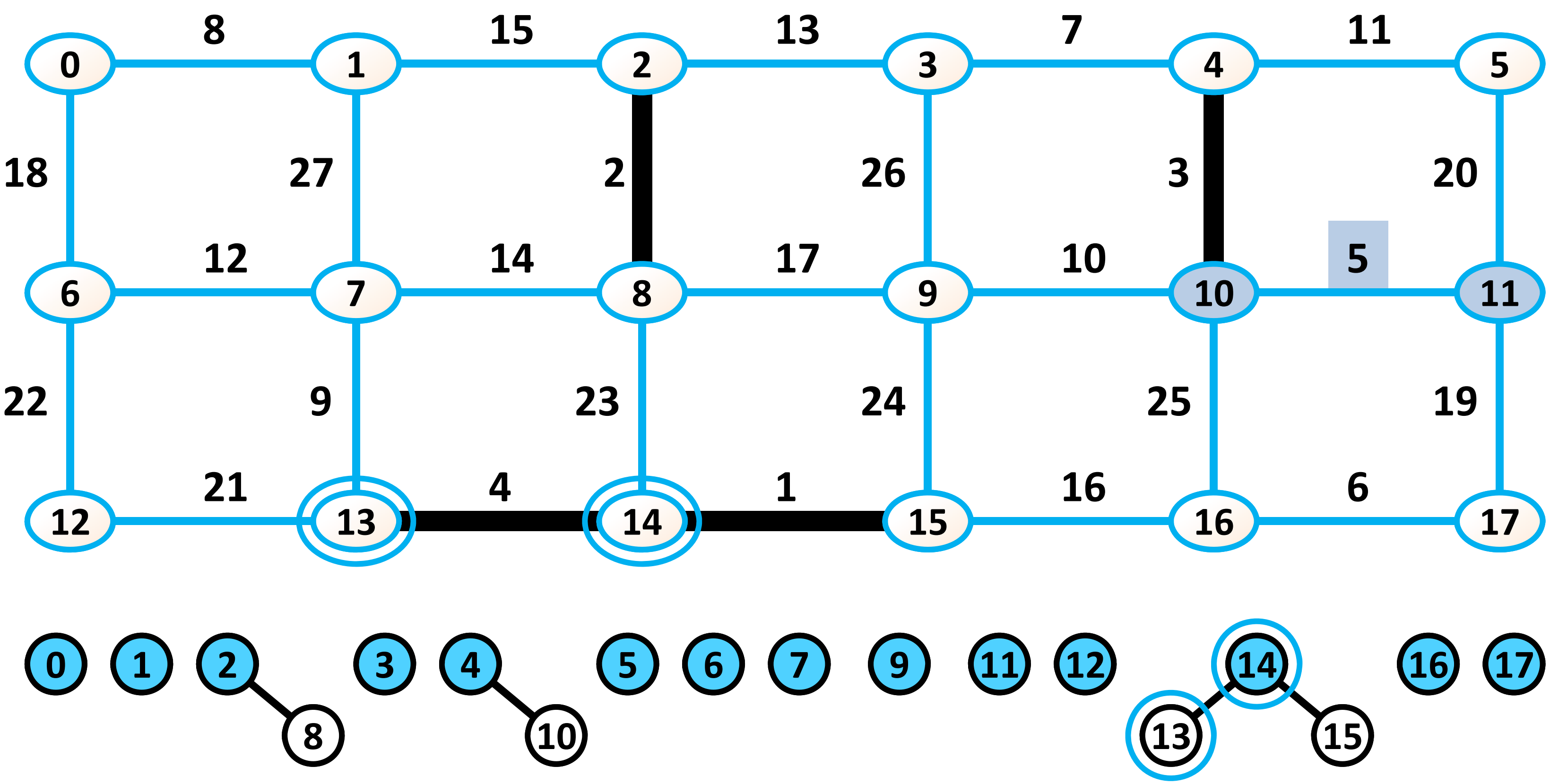
Kruskal's algorithm and Union-Find scheme example





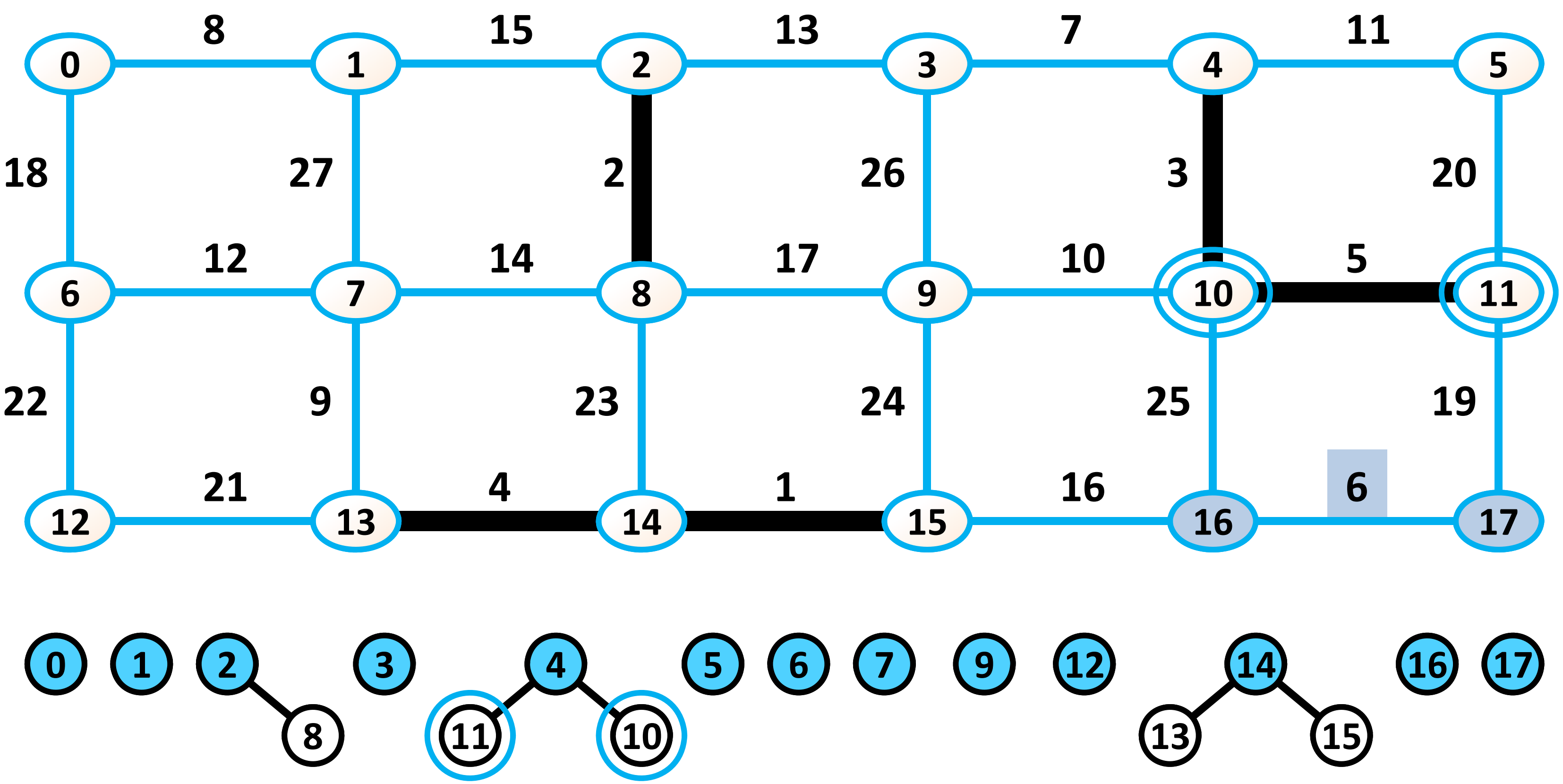
node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	1	2	3	4	5	6	7	2	9	4	11	12	13	14	14	16	17
rank	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0

Kruskal's algorithm and Union-Find scheme example



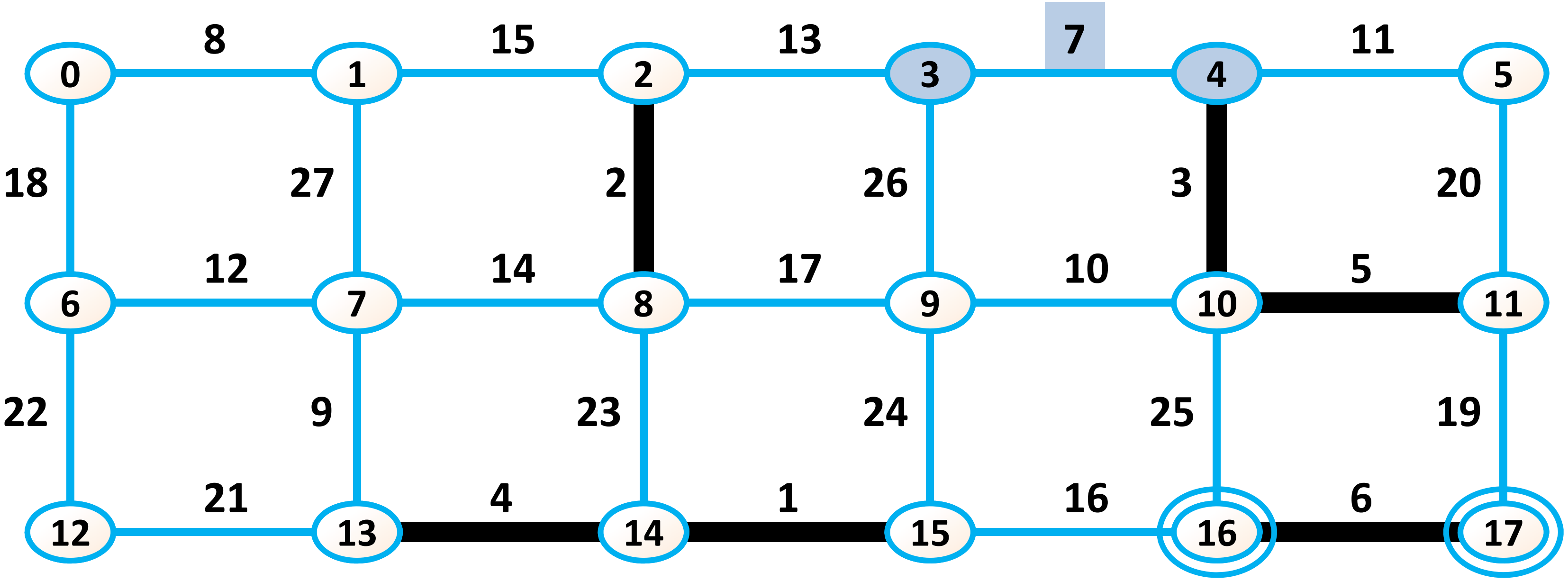
node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	1	2	3	4	5	6	7	2	9	4	11	12	14	14	14	16	17
rank	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0

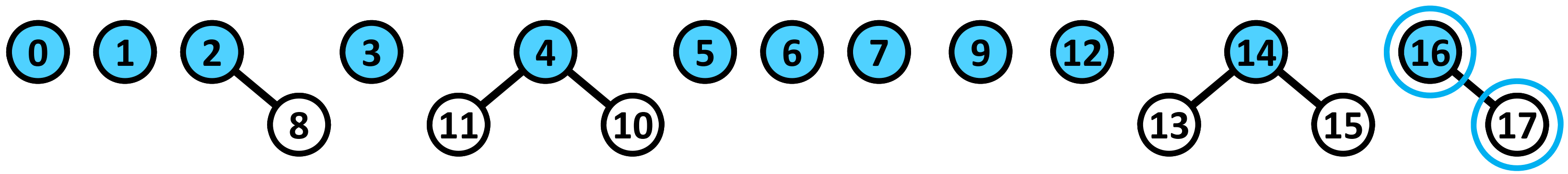
Kruskal's algorithm and Union-Find scheme example



node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	1	2	3	4	5	6	7	2	9	4	4	12	14	14	14	16	17
rank	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0

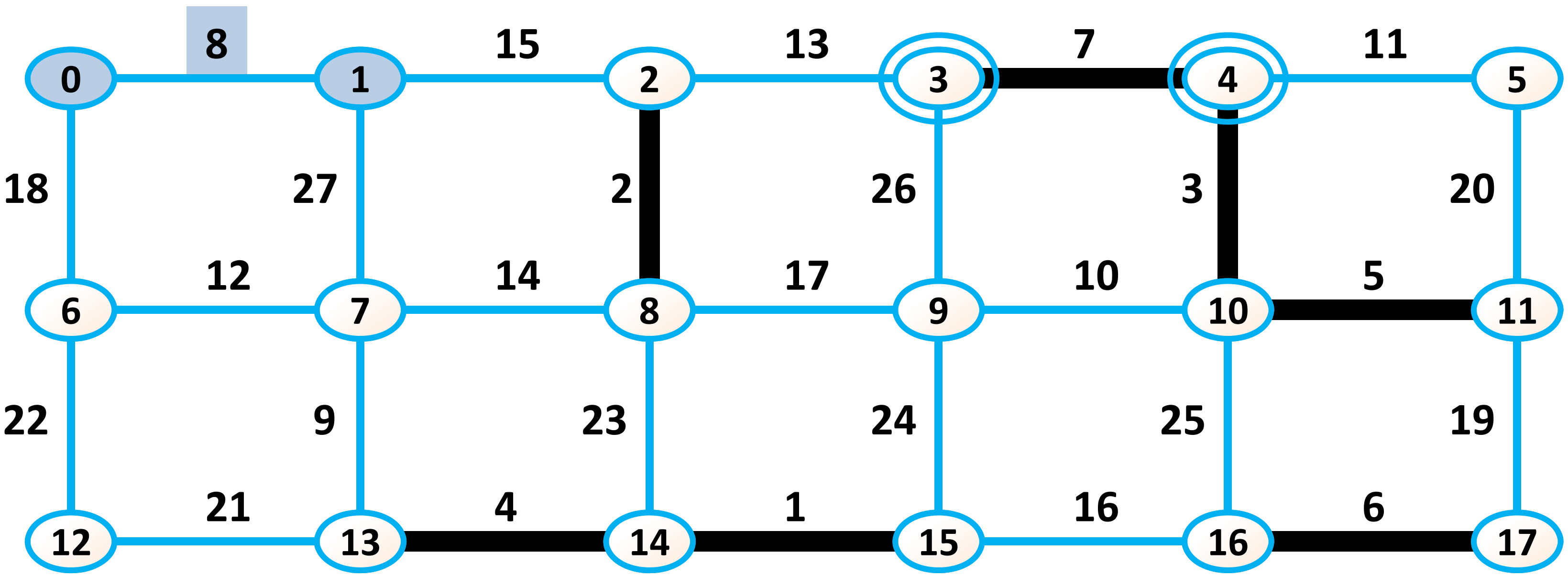
Kruskal's algorithm and Union-Find scheme example

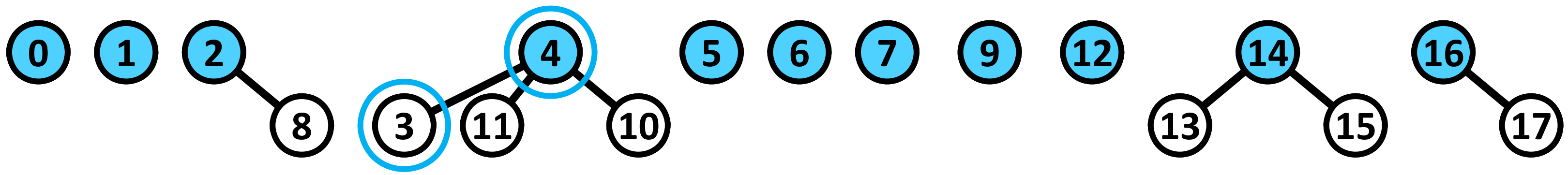




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	1	2	3	4	5	6	7	2	9	4	4	12	14	14	14	16	16
rank	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

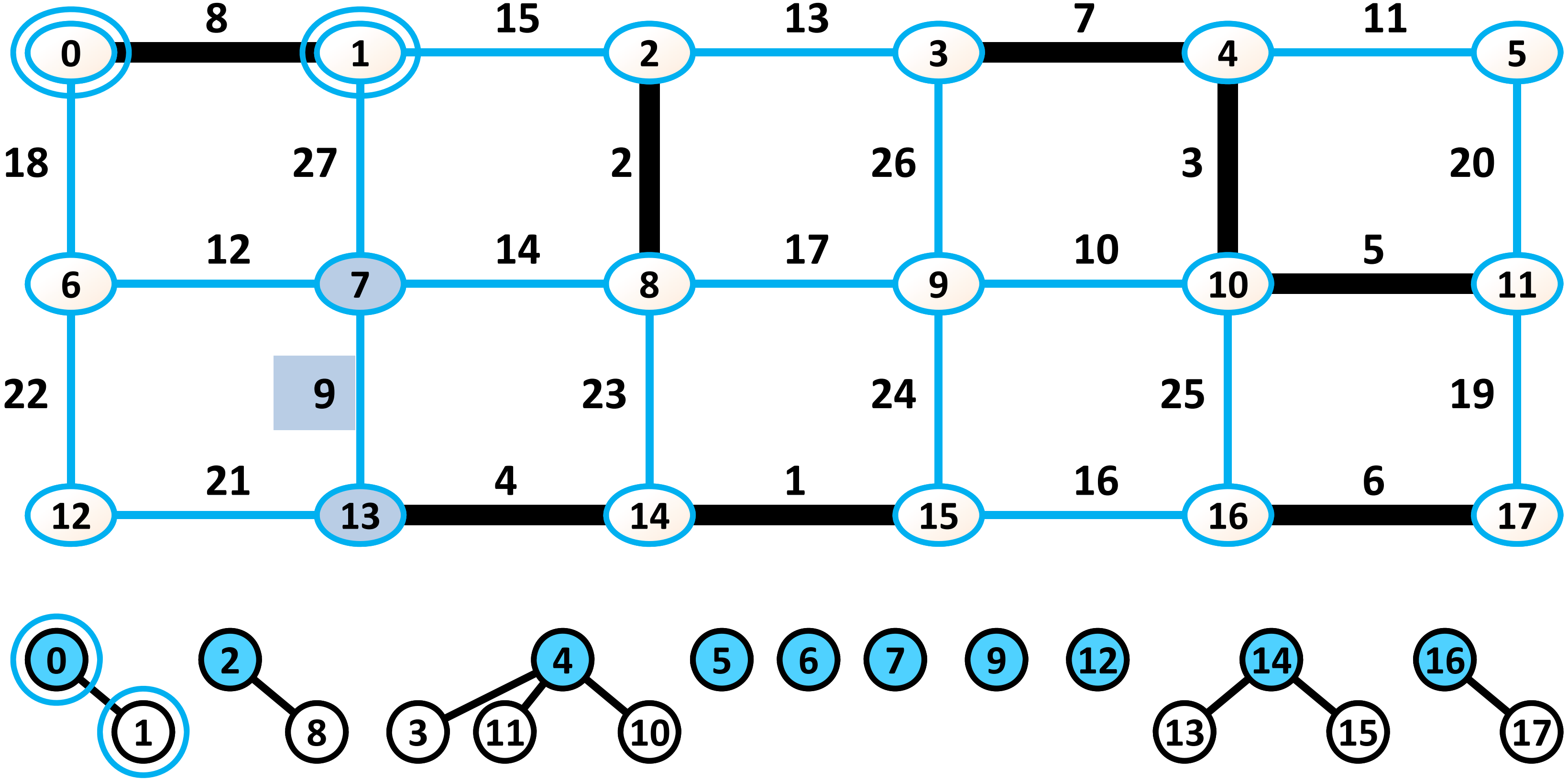
Kruskal's algorithm and Union-Find scheme example





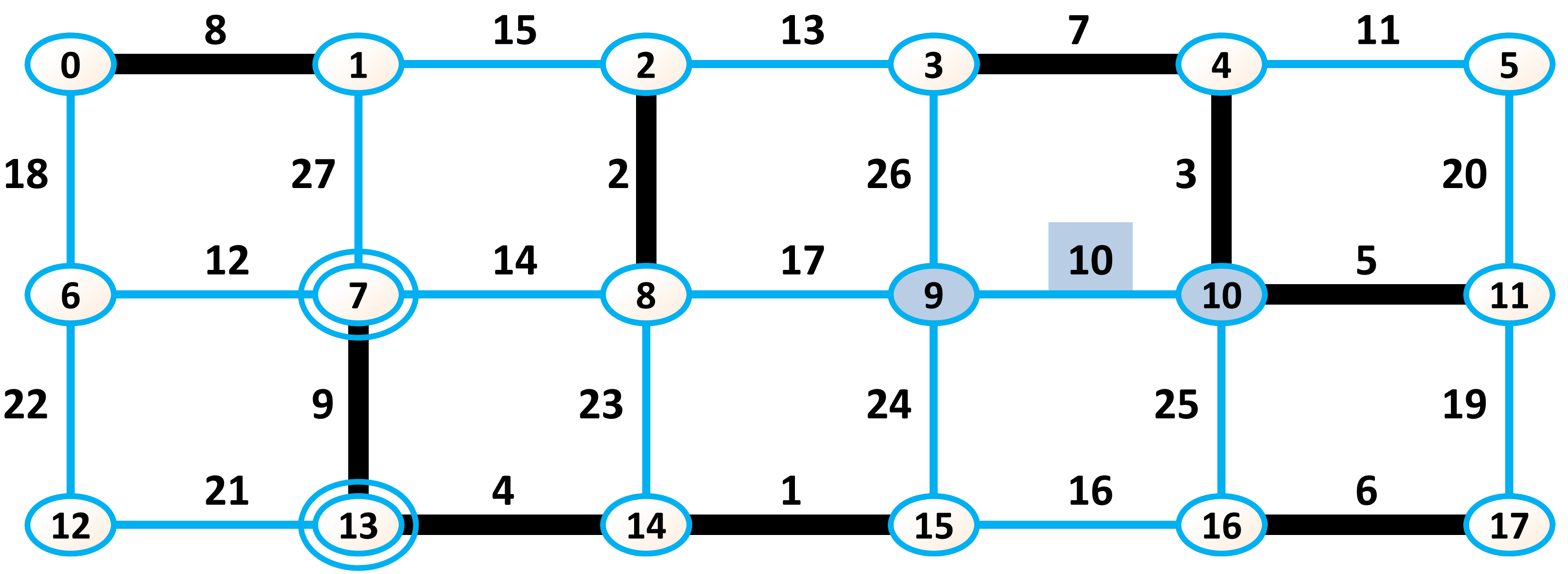
node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	1	2	4	4	5	6	7	2	9	4	4	12	14	14	14	16	16
rank	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

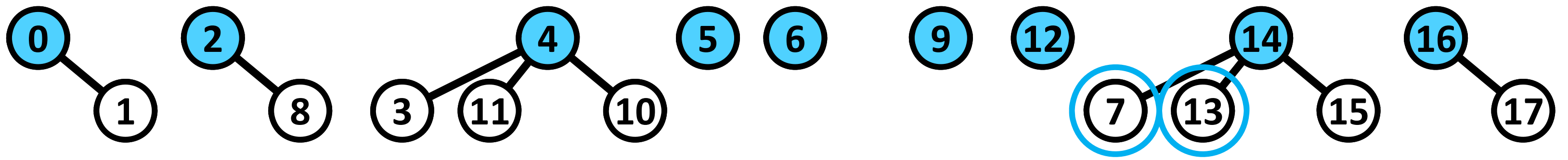
Kruskal's algorithm and Union-Find scheme example



node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	0	2	4	4	5	6	7	2	9	4	4	12	14	14	14	16	16
rank	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

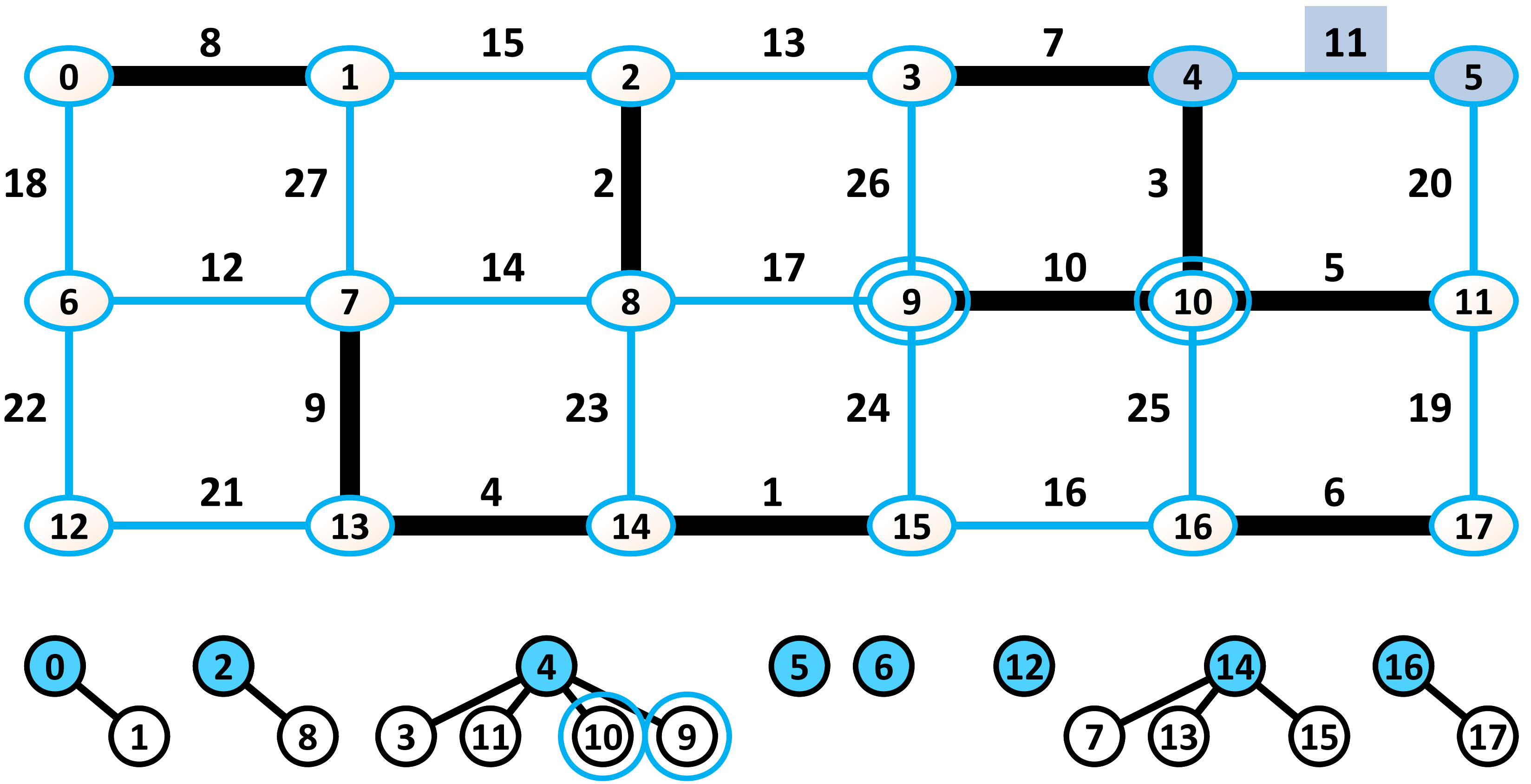
Kruskal's algorithm and Union-Find scheme example





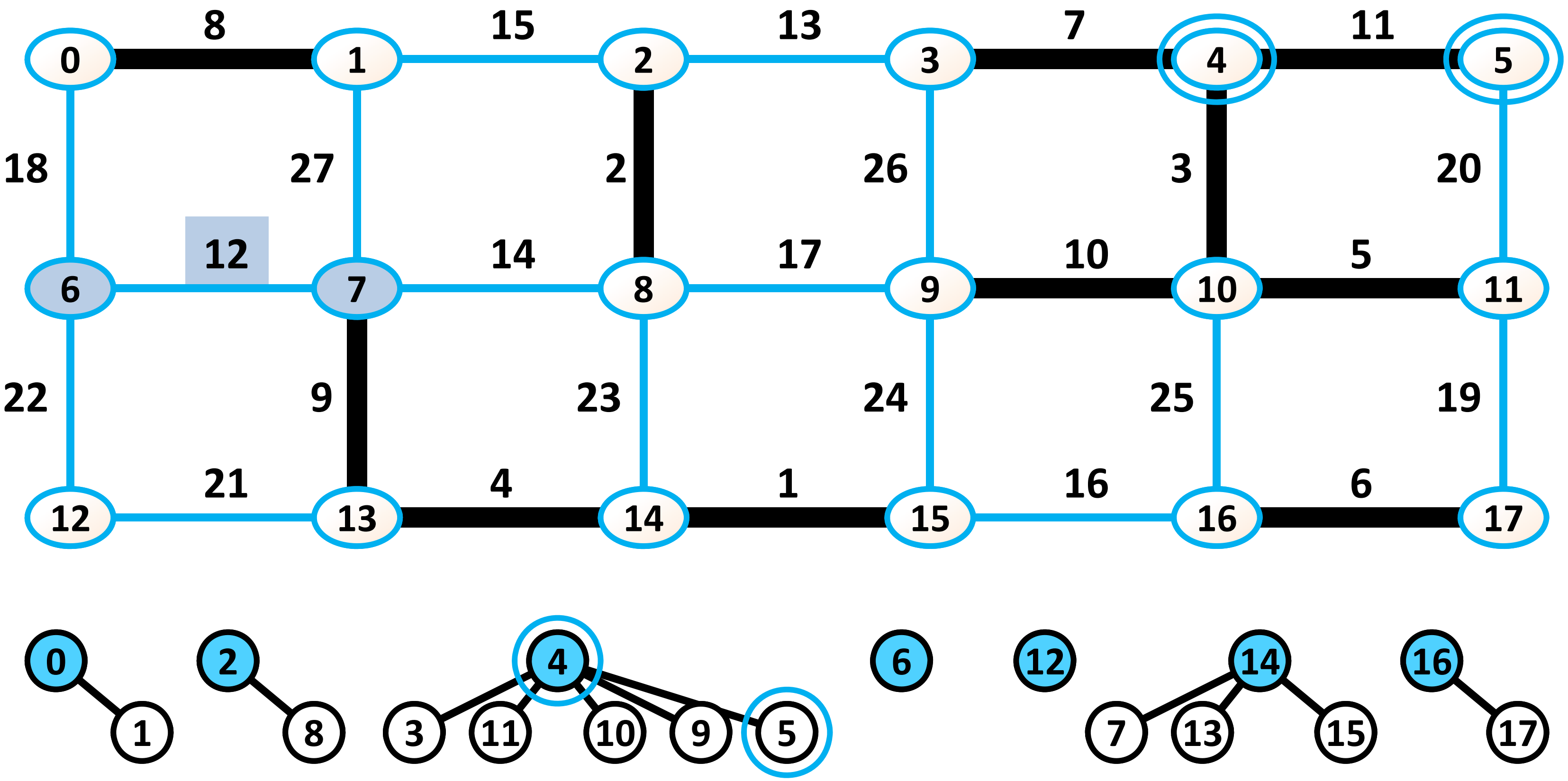
node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	0	2	4	4	5	6	14	2	9	4	4	12	14	14	14	16	16
rank	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

Kruskal's algorithm and Union-Find scheme example



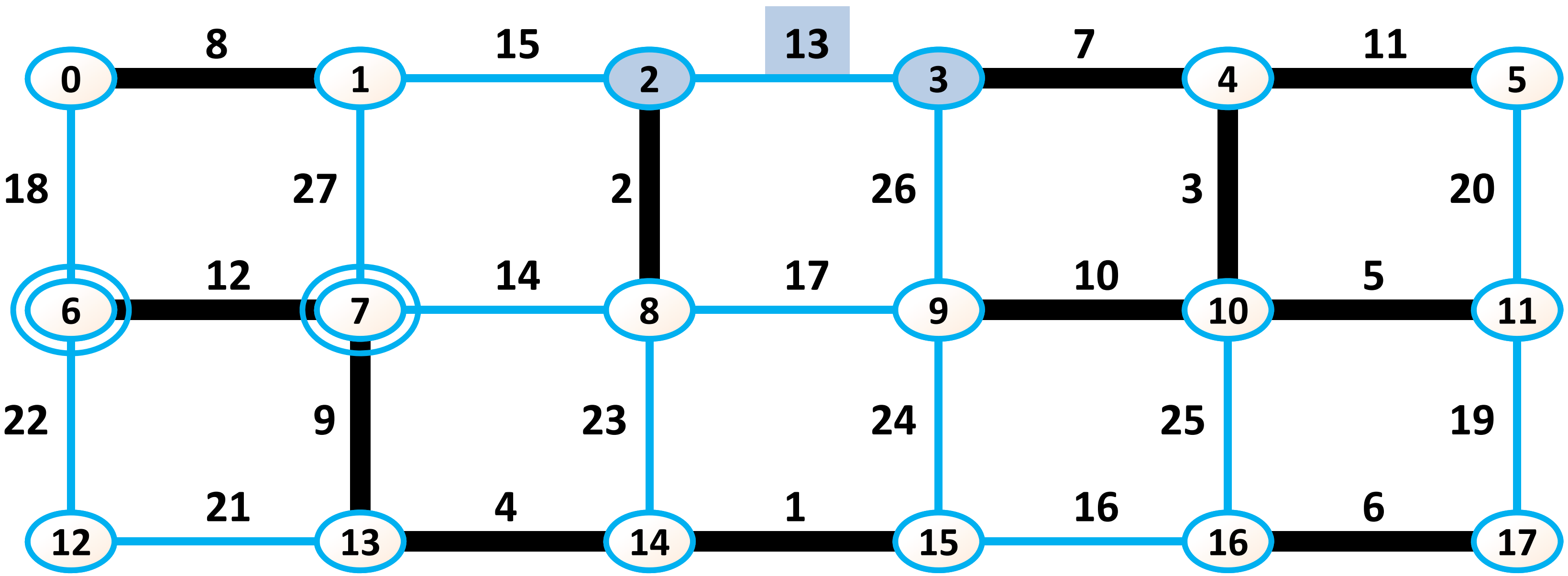
node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	0	2	4	4	5	6	14	2	4	4	4	12	14	14	14	16	16
rank	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

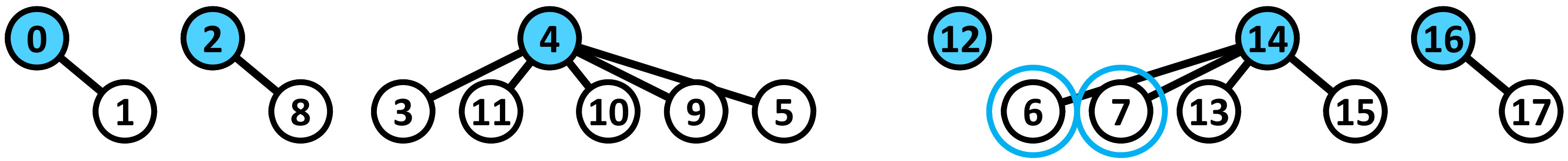
Kruskal's algorithm and Union-Find scheme example



node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	0	2	4	4	4	6	14	2	4	4	4	12	14	14	14	16	16
rank	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

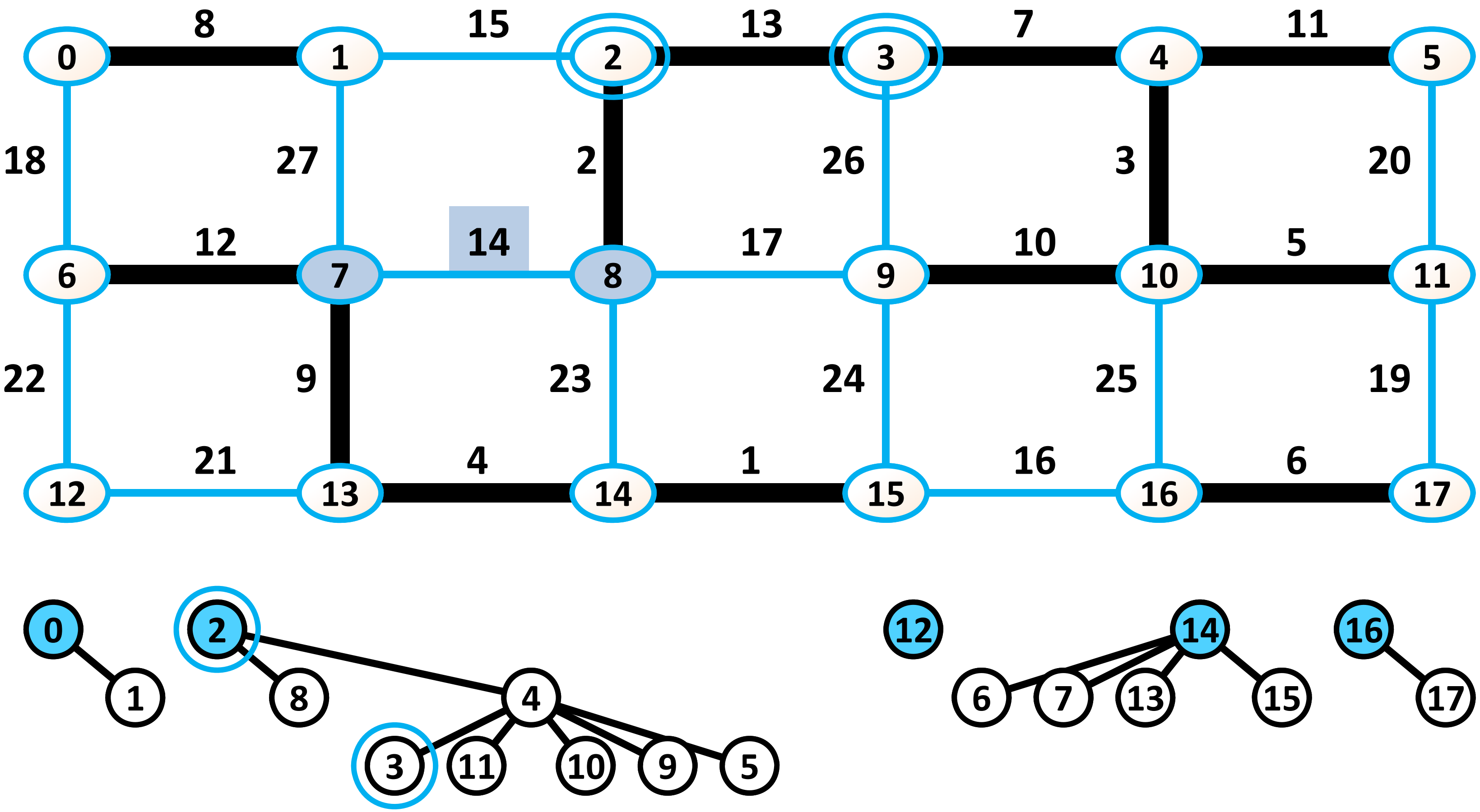
Kruskal's algorithm and Union-Find scheme example





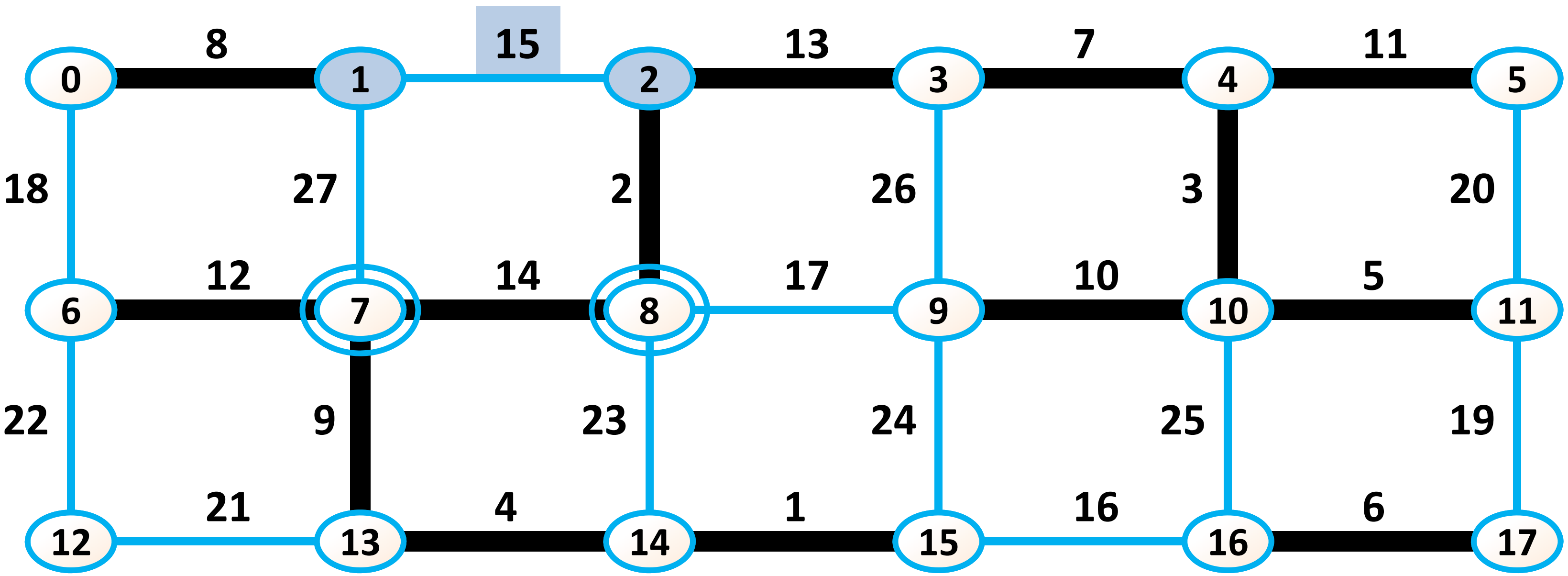
node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	0	2	4	4	4	14	14	2	4	4	4	12	14	14	14	16	16
rank	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

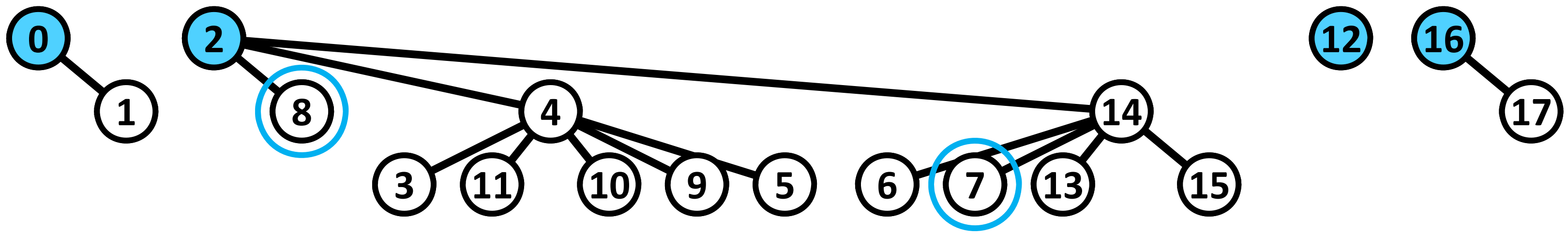
Kruskal's algorithm and Union-Find scheme example



node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	0	2	4	2	4	14	14	2	4	4	4	12	14	14	14	16	16
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

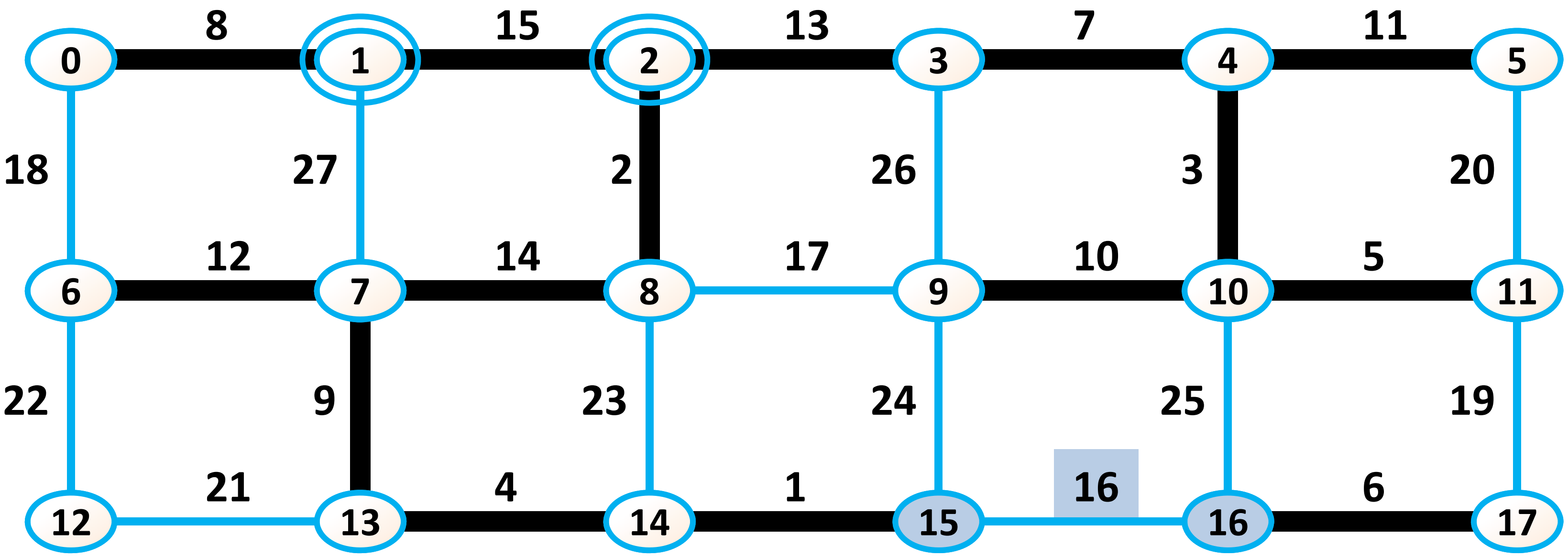
Kruskal's algorithm and Union-Find scheme example

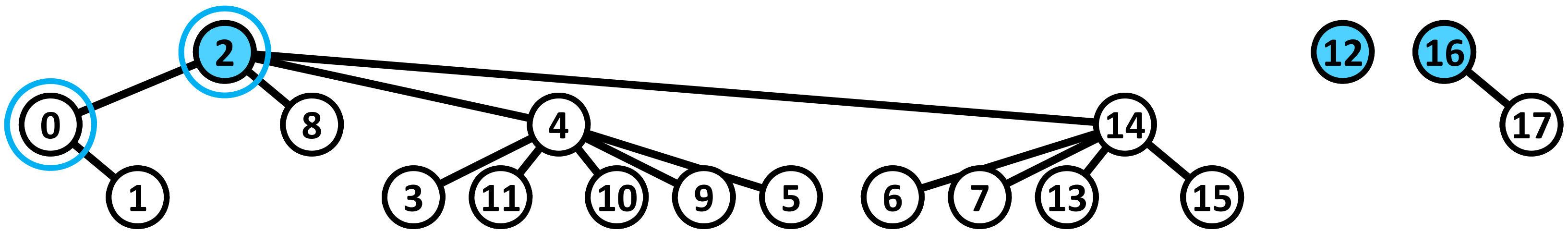




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	0	0	2	4	2	4	14	14	2	4	4	4	12	14	2	14	16	16
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

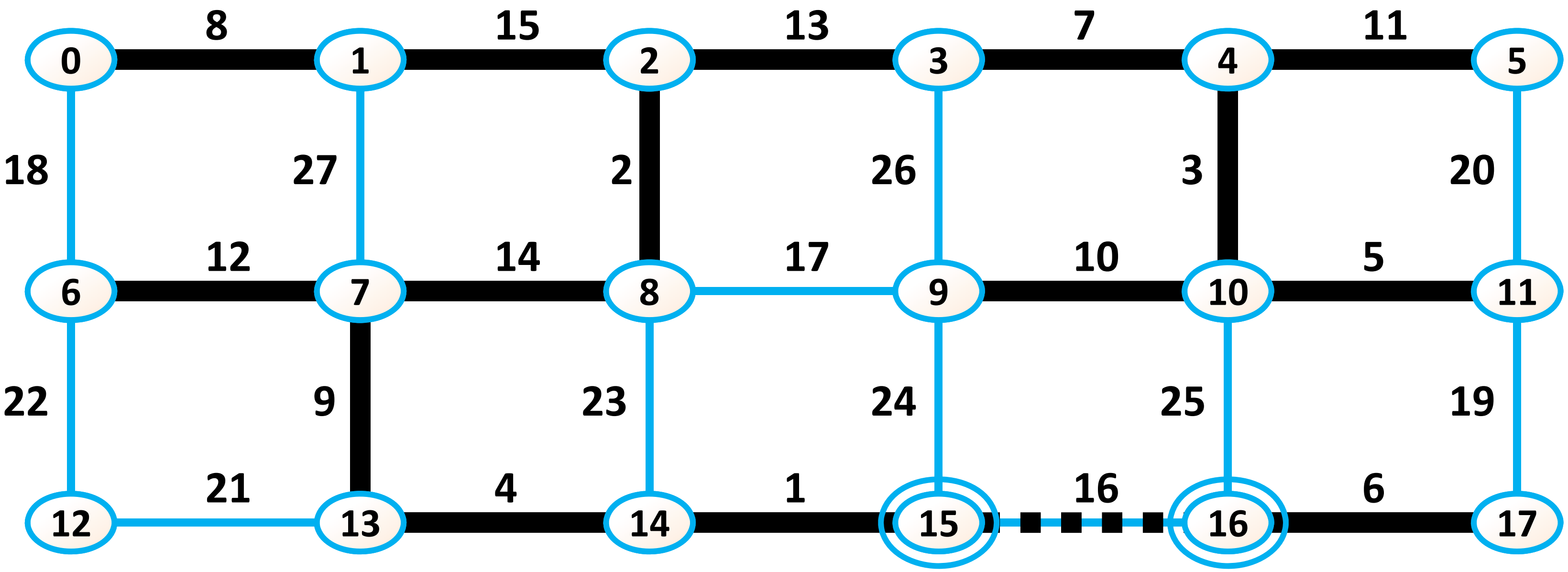
Kruskal's algorithm and Union-Find scheme example

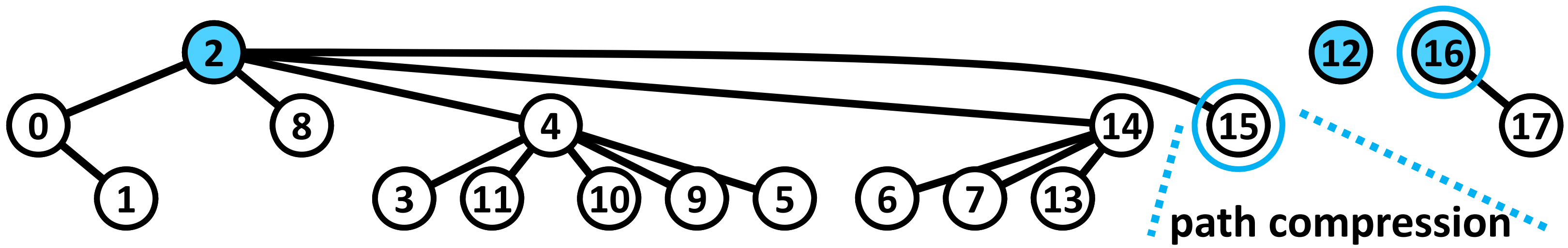




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	2	0	2	4	2	4	14	14	2	4	4	4	12	14	2	14	16	16
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

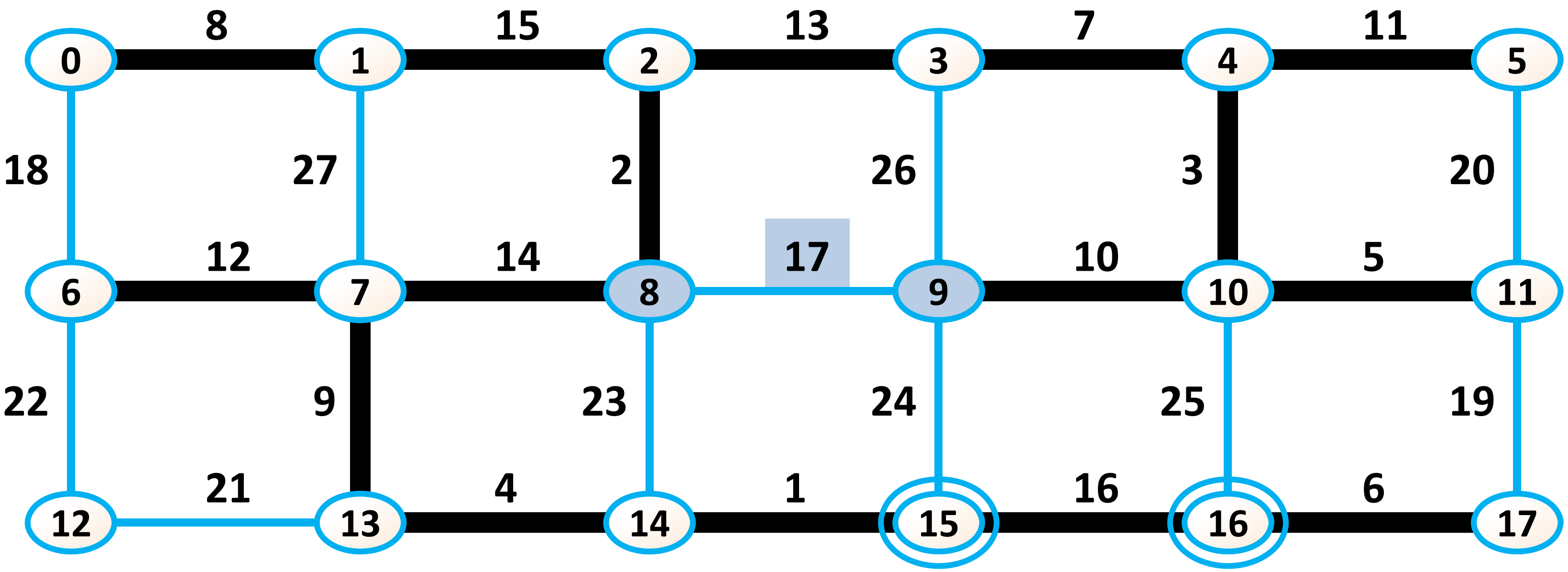
Kruskal's algorithm and Union-Find scheme example

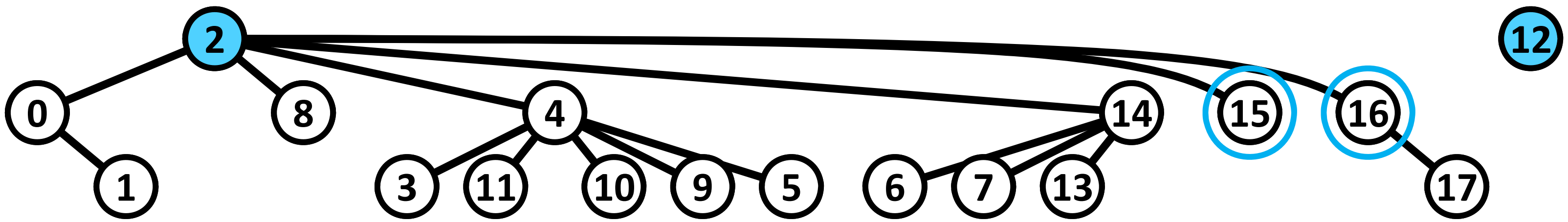




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	2	0	2	4	2	4	14	14	2	4	4	4	12	14	2	2	16	16
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

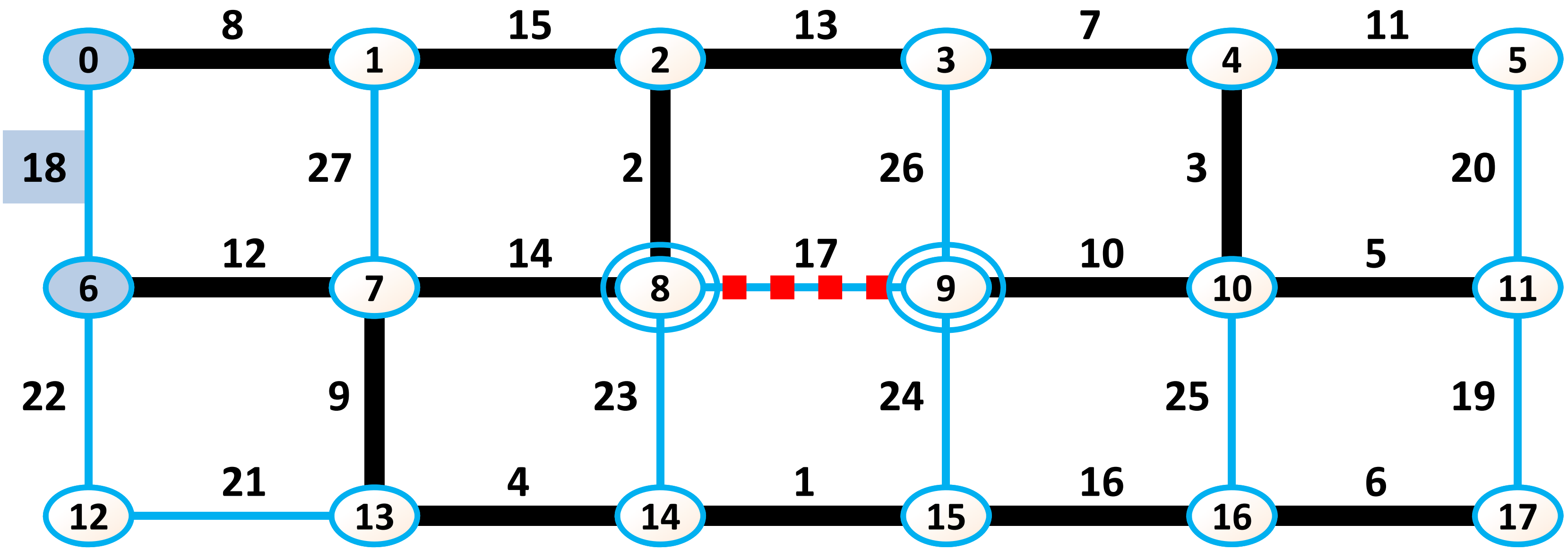
Kruskal's algorithm and Union-Find scheme example

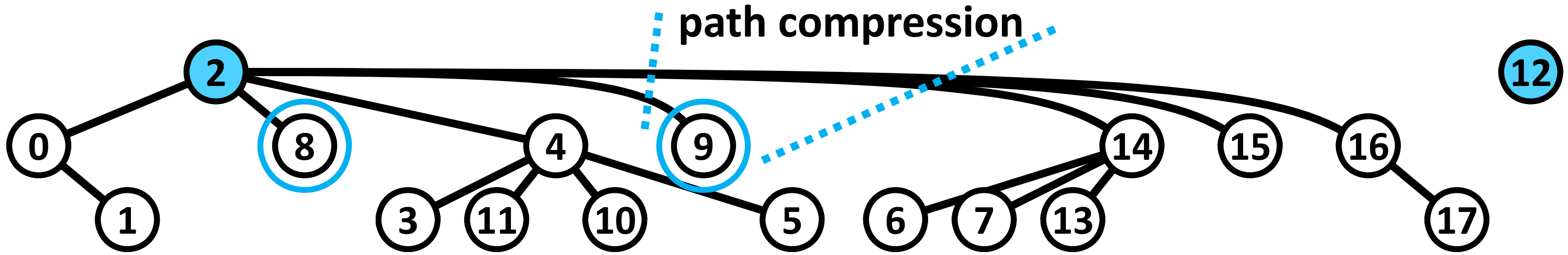




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	2	0	2	4	2	4	14	14	2	4	4	4	12	14	2	2	2	16
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

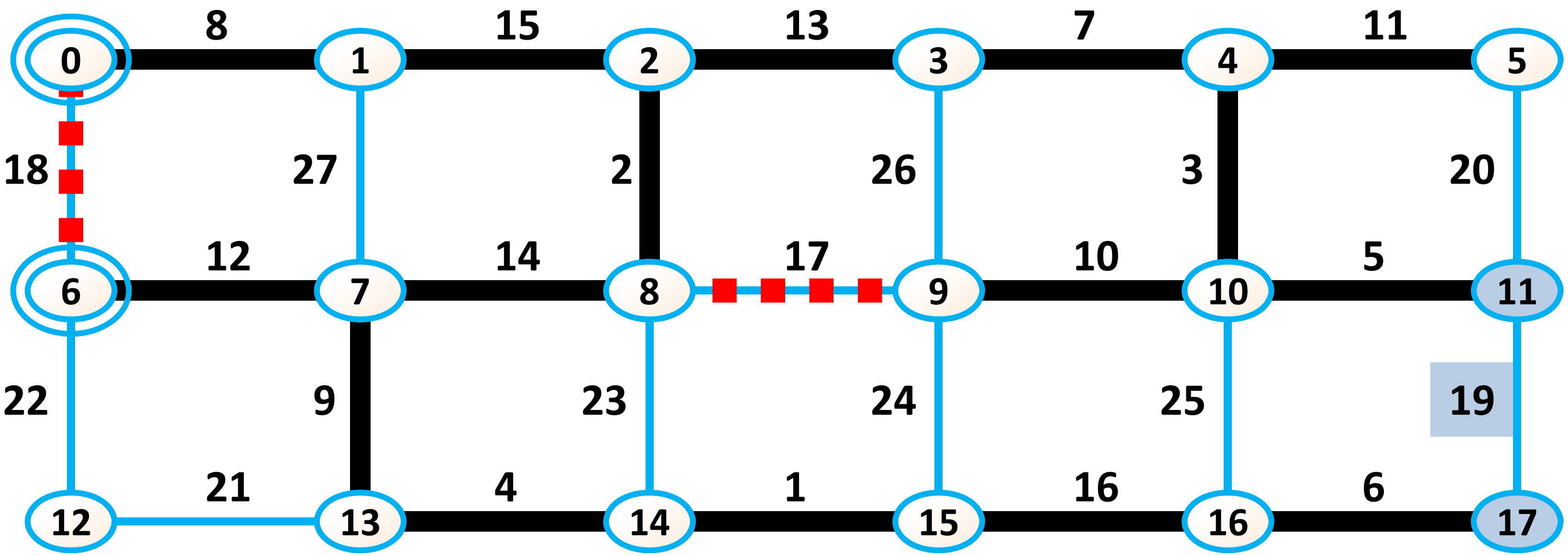
Kruskal's algorithm and Union-Find scheme example

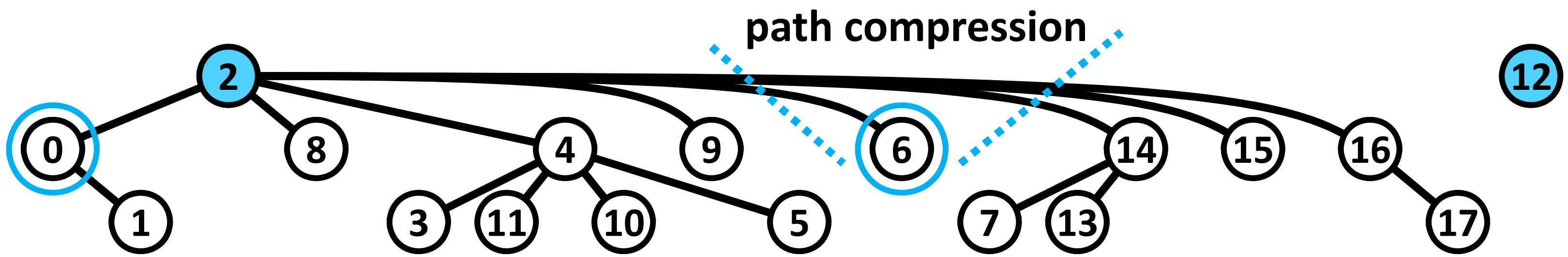




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	2	0	2	4	2	4	14	14	2	2	4	4	12	14	2	2	2	16
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

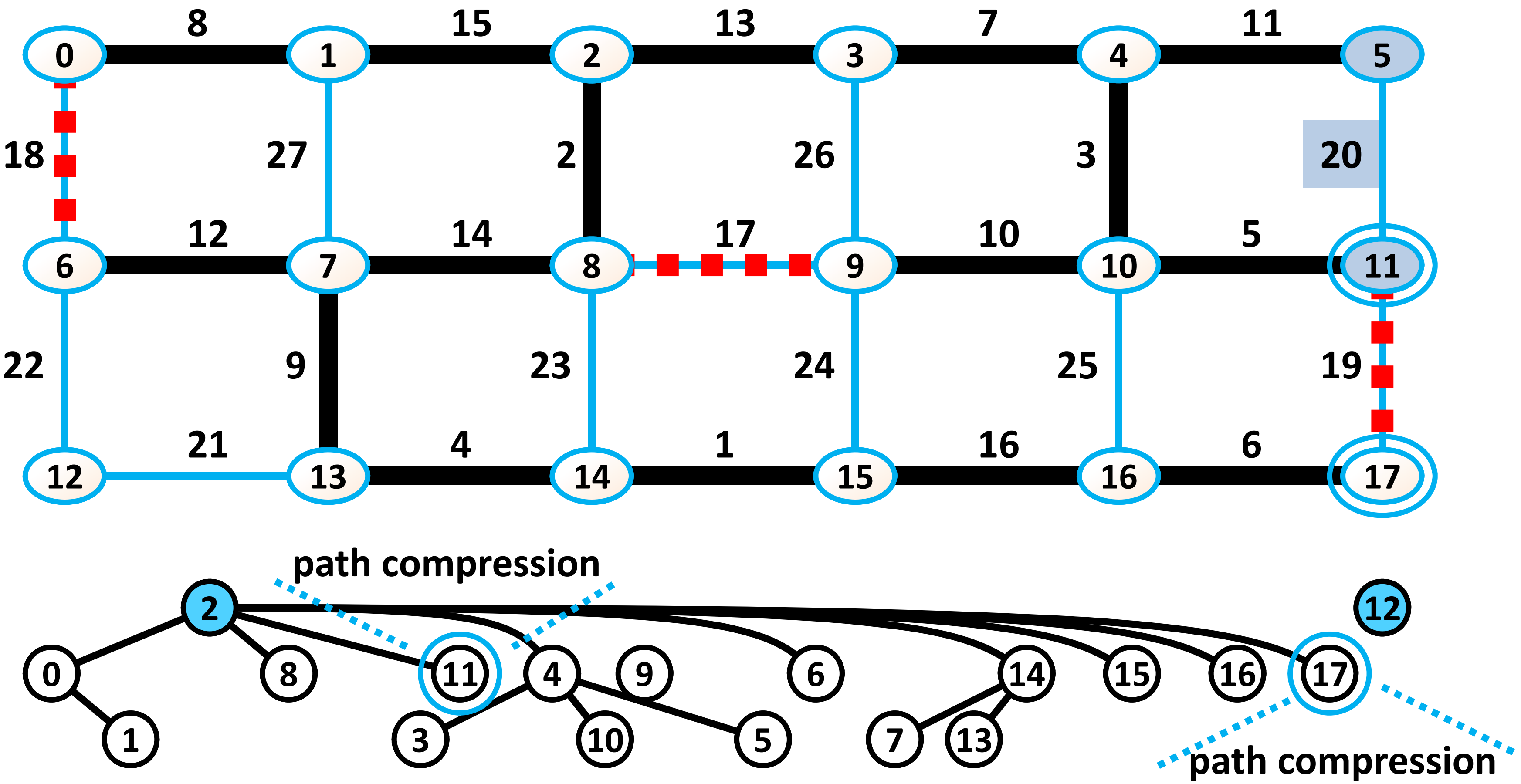
Kruskal's algorithm and Union-Find scheme example





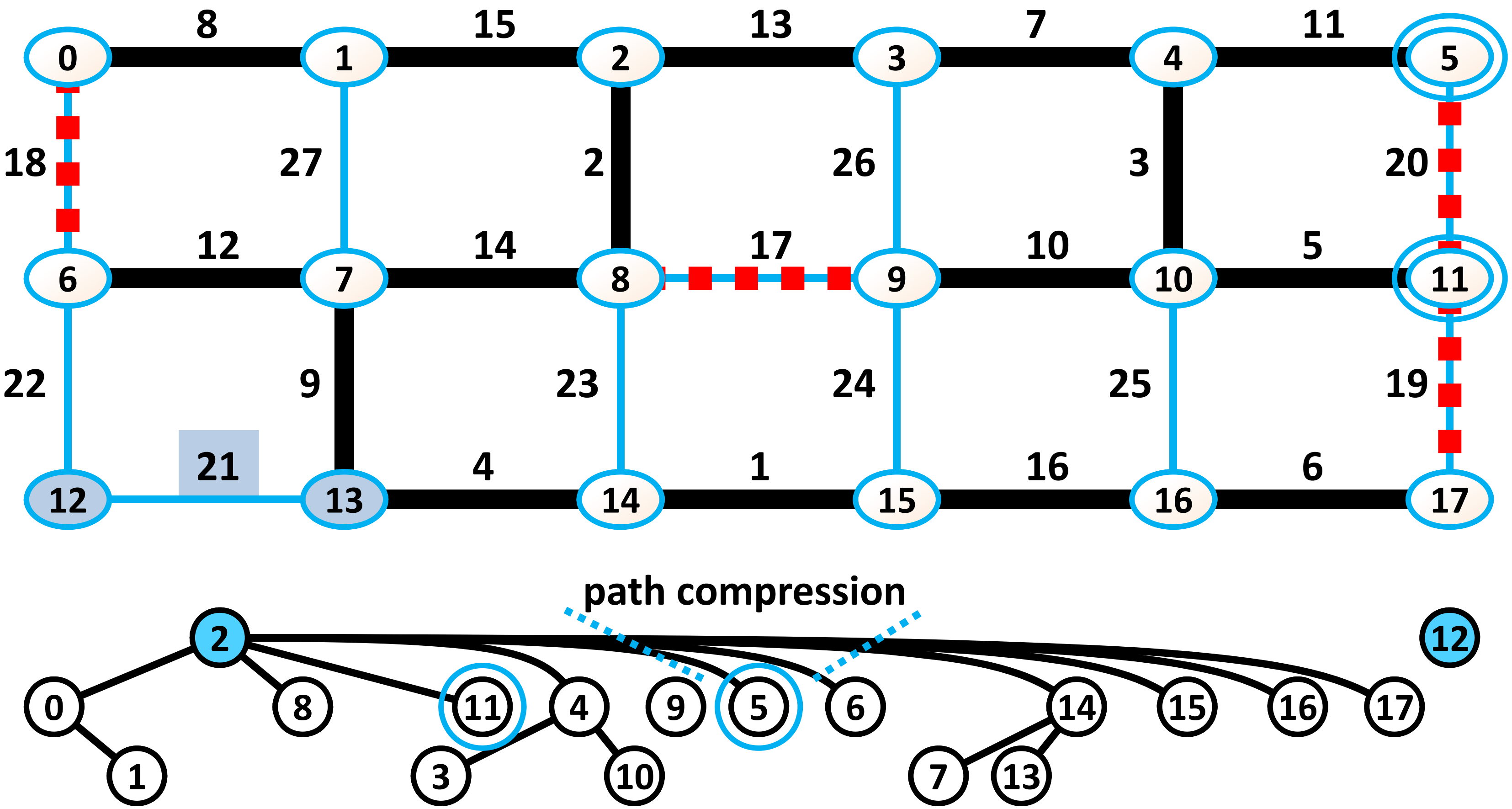
node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	2	0	2	4	2	4	2	14	2	2	4	4	12	14	2	2	2	16
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

Kruskal's algorithm and Union-Find scheme example



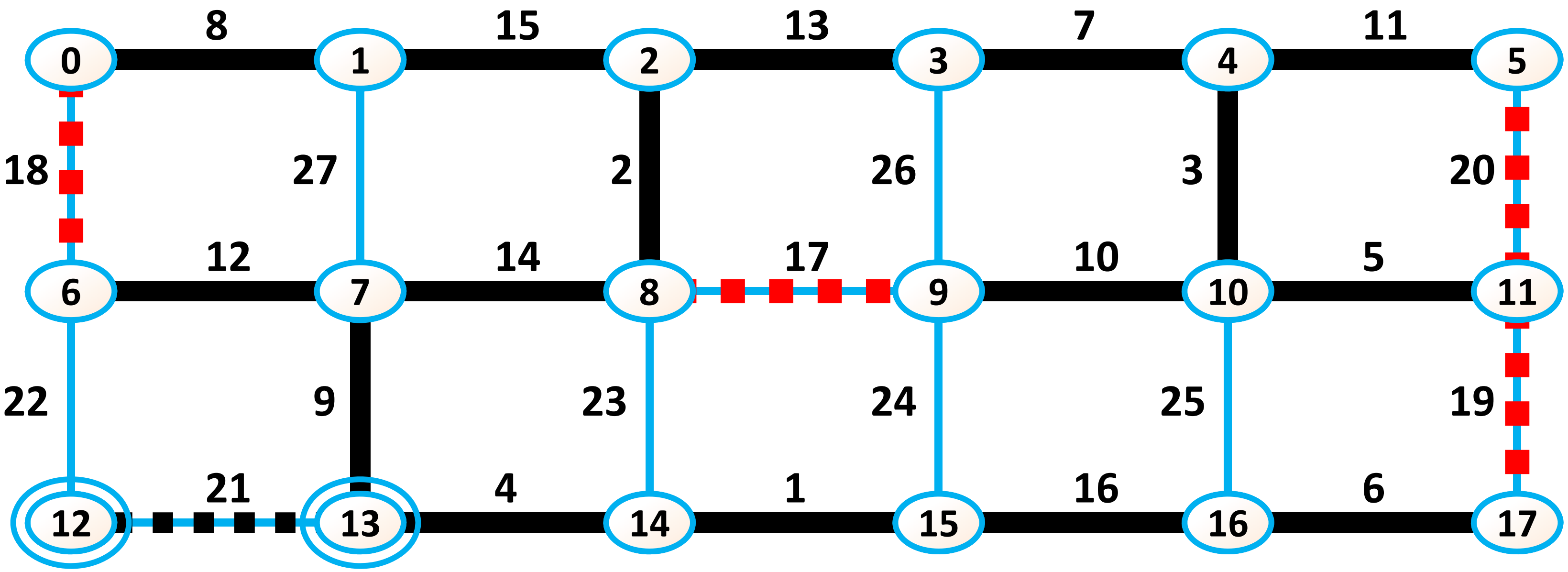
node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	2	0	2	4	2	4	2	14	2	2	4	2	12	14	2	2	2	2
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

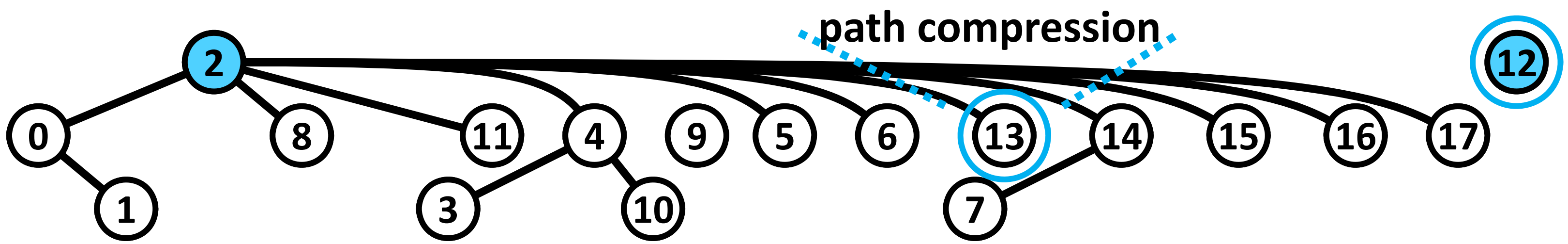
Kruskal's algorithm and Union-Find scheme example



node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	2	0	2	4	2	2	2	14	2	2	4	2	12	14	2	2	2	2
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

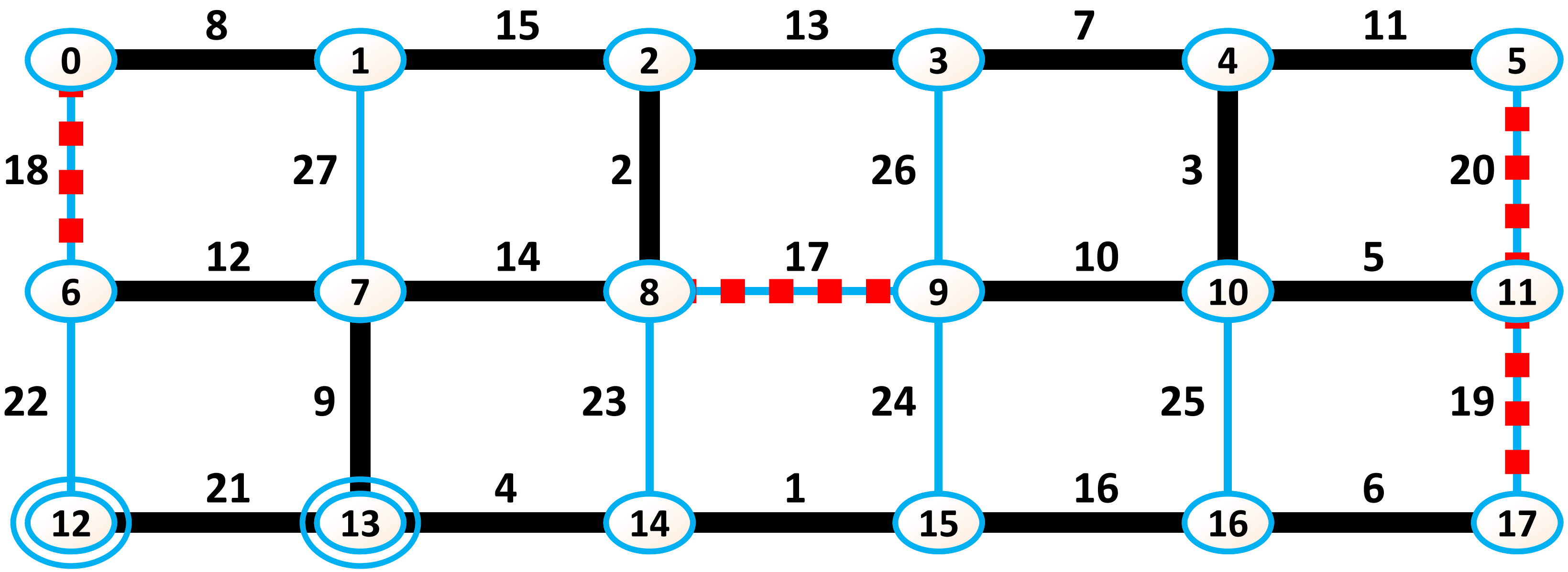
Kruskal's algorithm and Union-Find scheme example

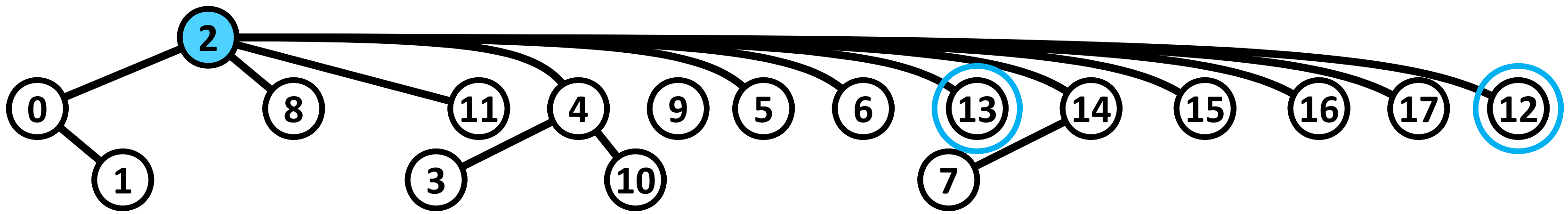




node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	2	0	2	4	2	2	2	14	2	2	4	2	12	2	2	2	2	2
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

Kruskal's algorithm and Union-Find scheme example





node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
boss	2	0	2	4	2	2	2	14	2	2	4	2	2	2	2	2	2	2
rank	1	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0

Kruskal's algorithm running time

When both union by rank and path compression are used, the running time spent on Union-Find operations in a graph with N nodes and M edges is $O(M \cdot \alpha(N))$, where $\alpha(N)$ is the inverse function of $f(x) = A(x, x)$, where $A(x, y)$, is the Ackermann function, known to grow quite fast.
In fact, for *any* graph representable in *any* conceivable machine $\alpha(N) < 4$.

Thanks to the inverse Ackermann function, all Union-Find operations run in amortized constant time in all practical situations.
It means that the speed bottleneck in Kruskal's algorithm is the initial edge sorting.
Sorting can be done in linear time when:

- edge values are strings (does not happen too often), apply **Radix sort**,
- edge values are integers in some (relatively) moderate range (e.g 0..10 000, etc.), apply **Counting sort**,
- edge values are floats (more or less) uniformly distributed over some interval, apply **Bucket sort**.

Conclusion
In many practical situations, a careful implementation of Kruskal algorithm may run in $\Theta(M)$ time.

Ackermann function

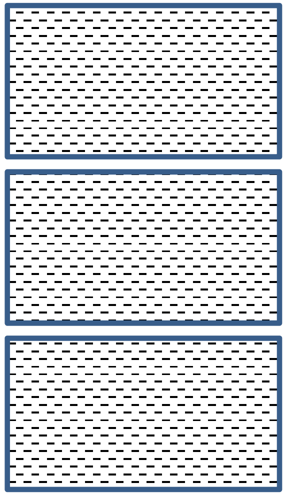
$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0, \\ A(m-1,1) & \text{if } m > 0, n = 0, \\ A(m-1, A(m, n-1)) & \text{if } m > 0, n > 0. \end{cases}$$

$m \backslash n$	0	1	2	3	4	5	n
0	1	2	3	4	5	6	$n+1$
1	2	3	4	5	6	7	$n+2$
2	3	5	7	9	11	13	$2n-3$
3	5	13 $= 2^4 - 3$	$= 2^5 - 3$ $= 29$	$= 2^6 - 3$ $= 61$	$= 2^7 - 3$ $= 125$	$= 2^8 - 3$ $= 253$	$2^{n+3} - 3$
4	13 $= 2^{2^2} - 3$	65533 $= 2^{2^{2^2}} - 3$	(#) $= 2^{2^{2^{2^2}}} - 3$	(##) $= 2^{2^{2^{2^{2^2}}}} - 3$	(###) $= 2^{2^{2^{2^{2^{2^2}}}}} - 3$	(####) $= 2^{2^{2^{2^{2^{2^{2^2}}}}}} - 3$	$\left.2^{2^{2^{\cdot^{\cdot^{2}}}}}\right\} n+3$ $- 3$
5	65533 $= 2^{2^{2^2}} - 3$	$\left.2^{2^{2^{\cdot^{\cdot^{2}}}}}\right\} 65536$ $- 3$	$\left.2^{2^{2^{\cdot^{\cdot^{2}}}}}\right\} \left.2^{2^{2^{\cdot^{\cdot^{2}}}}}\right\} 65536$ $- 3$				*too big to fit here …*
6	$A(6,0)$ $= A(5,1)$						*too big to fit here …*
7							

1. 0-th value in each line = 1st value in the previous line.
2. Value X in the j-th column (j > 0) on the current line is equal to the value on the previous line which column index is equal to the value written to left of the value X on the current line (= in the (j–1)-th column).

$A(4,2) = (\#) = 2^{2^{2^{2}}} - 3 = 2^{2^{4}} - 3 = 2^{2^{16}} - 3 = 2^{65536} - 3 =$

= (next 3 slides with 19729 decimal digits) =



$A(4,3) = (\#\#) = 2^{(\#)} - 3 = ?$

$A(4,4) = (\#\#\#) = 2^{2^{(\#)}} - 3 = ??$

Informal discussions concerning big integers:

http://waitbutwhy.com/2014/11/1000000-grahams-number.html
http://www.scottaaronson.com/writings/bignumbers.html

20035299304068464649790723515602557504478254755697514192650169737108940595563114530895061308809333481010382343429072631818229493821188126688
69506364761547029165041871916351587966347219442930927982084309104855990570159318959639524863372367203002916969592156108764948889254090805911
45703767520850020667156370236612635974714480711177481588091413574272096719015183628256061809145885269982614142503012339110827360384376787644
90432059603791244909057075603140350761625624760318637931264847037437829549756137709816046144133086921181024859591523801953310302921628001605
68670105651646750568038741529463842244845292537361442533614373729088303794601274724958414864915930647252015155693922628180691650796381064132
27530726714399815850881129262890113423778270556742108007006528396332215507783121428855167555407334510721311242739956298271976915005488390522
38043570458481979563931578535100189920000241419637068135598404640394721940160695176901561197269823378900176415171900511334663068981402193834
81435426387306539552969691388024158161859561100640362119796101859534802787167200122604642492385111393400464351623867567078745259464670903886
54774348321789701276445552940909202195958575162297333357615955239488529757995402847194352991354376370598692891375715374000198639433246489005
25431066296691652434191746913896324765602894151997754777031380647813423095961909606545913008901888875880847336259560654448885014473357060588
17090162108499714529568344061979690565469813631162053579369791403236328496233046421066136200220175787851857409162050489711781820400187282939
94344618622432800983732376493181478984811945271300744022076568091037620399920349202390662626449190916798546151577883906039772075927937885224
12943010174580868622633692847258514030396155585643303854506886522131148136384083847782637904596071868767285097634712719888906804782432303947
18650525660978150729861141430305816927924971409161059417185352275887504477592218301158780701975535722241400019548102005661773589781499532325
20858975346354700778669040642901676380816174055040511767009367320280454933902799249186730653993164072049223847481528061916690093380573212081
63507076343516698696250209690231628593500718741905791612415368975148082619048479465717366010058924766554458408383347905441448176842553272073
15586349347605137419779525190365032198020108764738368682531025183377533908861426184800374008082238104076468878471647552945326947661700424461
06331123802113458869453220011656407632702307429242605158281107038701834532456763562595143003203743274078087905628366340696503084422585596703
92718694611585137933864756997485686700798239606043934788508616492603049450617434123658283521448067266768418070837548622114082365798029612000
27441324438432402331257403545019352428776430880232850855886089962774458164680857875115807014743763867976955049991643998284357290415378143438
84730348426190338884149403136613985425763557710533558020662218557706008255128889333222643628198483861323957067619140963853383237434375883085
92337222846442879962456054769324289984326526773783731732880632107532112386806046747084280511664887090847702912081611049125555983223662448685
56651402684641209694982590565519216188104341226838996283071654868525536914850299539675503954938371853405900096187489473992880432496373165753
80367358671017578399481847179849824694806053208199606618343401247609663951977802144119975254670408060849934417825628509272652370989865153946
21930046073645079262129759176982938923670151709920915315678144397912484757062378046000099182933213068805700465914583872080880168874458355579
26258465124763087148566313528934166117490617526671492672176128330845273936469244582892571388877839056300482483799839692029222215486145902373
47822268252163995744080172714414617955922617508388902007416992623830028228624928418267124340575142418856999427233160699871298688277182061721
44531425749440150661394631691976291815065797455262361912248480638900336690743659892263495641146655030629659601997206362026035219177767406687
77463549375318899587866282125469797102065747232721372918144666659421872003474508942830911535189271114287108376159222380276605327823351661555
14936937577846667014571797190122711781278045024002638475878833939681796295069079881712169068692953824852983002347606845411417813911064856023
65497542274972310076151318700240539105109138178437217914225285874320985249578780346837033378184214440171386881242499844186181292711985333153
82567321870421530631197748535214670955334626336610864667332292409879849256691109516143618601548909740241913509623043612196128165950518666022
03071561368473236466086890501426391390651506390819937885231836505989729912540447944342516677429965981184923315155527288327402835268844240875
28112832899806259126736995462473415433335001472314306127503903073971352520693381738433229507010490618675394331307847980156551303847581556852
36218010419650255596181934986315913233036096461905990236112681196023441843363334594927631946101716652913823717182394299216272538461776065694
54229787707138319881703696458868981186321097690035573588462446483570629145305275710127887202796536447972402540544813274839179412882642383517
19491972097971459368875371987291308317380339110161285474153773777159517280841116275971863849242228023734419254699919836721921312870355853079
66942713416391033882754318613643490100943197409047331014476299861725424423355612237435715825933382804986243892498222780715951762757847109475
11903348224141202518268871372819310425347819612844017647953150505711072297431456991522345164312184865757578652819756484350895838472292353455
94645212158316577514712987082259092926556388366511206819438369041162526687100445602437042006637090019411855571604720446436969328500600469281
40507119069261393993902735534545567470314903886022024639948260501762431969305640666366626090207048887438898907498152865444381862917382901051
82086993638266186830391527326458128678280660133750009659336462514609172318031293034787742123467911845479131110989779464821692250562939995679
34838016991574397005375421344858745868560472867510654233418938390991105864655951136460610551568385412174598018071331636125730796111683438637
67667307354583494789788316330129240800836356825939157113130978030516441716682518346573675934198084958947940983292500086389778563494693212473
42610306271374507728615692259662857385790553324064184901845132828463270926975383086730840914224765947443997334813081098639941737978965701068
70267341619671965915995885378348229882701256058423655895396903064749655841479813109971575420432563957760704851008815782914082507777385597901
291294073094627859445058594122731948127532251523248015034665190482289614066468903051025109162377704484862302294888966711380555607956620732449

3733740278367673002030116152270089218435156521213792157482068593569207902145022771330999877294595969528170445821819560809658117027980626 6989
1205061560742325686842271306295009864421853470810407128917646906550836129916694778023822502789667843489199409657361704586786242554006942 5166
9397929262471452494540885842272615375526007190433632919637577750217600519580069384763578958687848953687212289855780682651819270363209948 0155
8744555751753127364714212955364940843855866152080121150790750685533444892586932838596530132720469706945715469593536585717888948623332924 6520
2735853188533370948455403336565356988172582528918056635488363743793348411845580168331827676834646291995605513470039147876808640322629616 6415
6066750815371064672310846196424753749055374480531822600271021640098058449752602303564003808347205314994117296573678506642140084269649710 3241
9191821212132069397691439233683747092282677387081322366800869247034915868409911530983154120635661231875043054675369832308279664574176208 0659
3177265685841681837966106144963432544111706941700222657817358351259821080769101961052229263879745049019254311900620561906577452416191913 1875
3398404934397682331029846589331837301580959252282920682086223033258528011926649631444131644277300323779227471233069641714994553226103547 5145
6312906688543454268697884477429817774937101176146516241836166802548152963353084908499430067636548061029400946937506098455885580439704859 1444
9584445079978497045583550685408745163316464118083123079704389849190506587586425810738422420591191941674182490452700288263983057950057341 7114
8703118714283418449915345670291528010448514517605530697144176136858238410278765932466268997841831962031226242117739147720800488357833356 9204
5339359532545648970285585897355057512351295365405028420810227852487766035742463666731486802794860524457826736262308529782650571146248465 9591
4210278122788941448163994973881884622768244851622051817076722169863265701654316919742651230041757329904473537672536845792754365412826553 5818
5804684006936771860502007054724754840080553042495185449526724726134731817474218007857469346544713603697588411802940803961674694628854067 9172
1386012254195038197045384172680063988206563287928395827085109199588394482977756471520261328710895261634177071516428994879535648545535531 4875
4978134009964854498635824847690590033116961303766127923464323129706628411307427046202032013368350385425360313636763575212604707425311209 2334
0283748294945310472741896928727557202761527226828337674139342565265328306846999759709775000556088993268502504921288406827413988163154045 6490
3507758716800740556857240217586854390532281337707074158307562696283169556874240605277264858530506113563848519659189686495963355682169754 3762
1430778665934730450164822432964891270709898076676625671517269062058815549666382573829274182082278960684488222983394816670984039024283514 3068
1376725346012600726926296946867275079434619043999661897961192875051944235640264430327173734159128149605616835398818856948404534231142461 3559
9252723300648816274667235237512343118934421188850850793581638489944875447563316892138696755743027379537852625423290248810471819390372206 6689
4702204258836895840939998453560948869946833852579675161882159410981624918741813364726965123980677561947912557957446471427868624053750576 1042
0426714936608498023827468057598259133100691994190465190653117190892607794911921794640735512963386452303567334558803331319708036545718479 1550
4326548995597058628882868666066180218822486021449999731221641381706534801755104384066244128228036166489042573776409563264828252584076690 4560
8439490325290526337532316509087681336614242398309530806549661879381949120033919489494065132398816642080088395554942237096734840072642705 7011
6508907519615537018626479745638118785617545711340047381076276301495330973517418065547911266093803431137853253288353335202493436597912934 1284
8549709468263290758301930726653377825593143311109638480539408592839889077962104798479196868765399874770959127887274758744398067798249682 7827
2200926449944559380414608770641941810440758269805688038949654616587983904660587645341810289907194293021774519976104495043196841503455514 0448
2092893337865736305283061999007774872692299860827905317169187657886090894181705799340489021844155979109267686279659758395248392673488363 4745
6516870161662406424242412289611180106156823425393921800524834547237792199112285959141918774917938233400100781283265067102817813960291209 1472
0100947878752551263372884222353869490067927664511634758101193875319657242121476038284774774571704578610417385747911301908583877890152334 3430
1300528279703858035981518292960030568261209195094373732545417105638388704752895056396102984364136093564163258940813798151169333861979733 9821
6707610046079800960160248230969430438069566201232136501405495862506152825880330229083858124784693157203232336018994694376477267218793768 2643
1828382603564520699468630216048874528424363593558622333506235945002890558581611275341783750455936126130852640828051213873177490200249552 7387
3458595640516083058305377073253397155262044470542957353836111367752316997274029294167420442324811387507563131907827218886405337469421384 2169
9288629404796353051505607881263662064972312575790195988730411956262273437289005165611109411174527796548279047125058199907749806382155937 688
5546498822938985408291325129076478386322494781016753491693489288104203015610283386143827378160946341335383578340765314321417150655877547 8202
5245478065730134227747061674424196895261316427410469547462148375628829977180418678508454696561915090869587425118443583730659095146098045 1247
4094113738999278224929833677960110153870961297497055663016373072027507347599229437923938244274211861582361613178863925530951171884212985 0830
7238259729144142251579403883011359083331651858234967221259621812507058113759495525022747274674369887131926670769299199084467161228738858 4575
8462272657333075373557282395161696417519867501268174542932373829414382481437713986190671665757294580780482055951188168718807521297183263 6442
1553367877512747669407901170575098195750845635652173895441798750745238544552001335720333237989507439390531291821225525983379090946363020 218
5353848854825062897715616963860712382771725621313460549401770413581731931763370136332252819127547191443450920711848838366818174263342949 6118
7009150304916533946476371776643912079834749462739782217150209067019030246976215127852195614207080646163137323651785397629209202550028896 2012
9701413796400380557349492690735351459612086747965477336929587736286356601437679640384307968641385634478013282612845891848985280480488441 8082
1639423974014362903481665458114454366460032490618763039502356402044530748210241366895196644221339200757479128683805175150634662569391937 7402

83512075666260829890491877287833852178522792045771846965855278790447562192663992008409302075673925363735628390829817577902153202106409617373
28359849406665214119818381088451545977289516457213189779790749194101314836854463961690460703010759681893374121757598816512700076126278916951
04063158576375347874200702220510708912576123616580268068158584998526314658780866168007332646768302063916972030648944056281954061906852420030
53463156621891327309069687353181641094514288036605995220248248886711554429104721929134248346438705368508648749099178812670565665387191049721
82004237149274016446094345984539253670613221061653308566202118896823400575267548610147699368873820958455221157192347968688816085363161586288
01503959494185294892270744108282071693033878180849362040182552222710109856534448172074707560192459155994310729495781978785905789400525401228
67517142511184356437184053563024181225473266093302710397968091064939272722683035410467632591355279683837705019855234621222858410557119921731
71796980433931770775075562705604783177984444763756025463703336924711422081551997369137197516324130274871219986340454824852457011855334267526
47159783107312456634298052214554941562527240289153333543493412178620370072603152798707718724912344944771479095207347613854254853115527733010
30342476835865496093722324007154518129732692081058424090557725645803681462234493189708138897143299831347617799679712453782310703739151473878
69211918756670031932128189680332269659445928621060743882741691946516226763254066507088107103039417886056489376981673415902592519461182364294
56526693722031555047002135988462927580125277154220166299548631303249123110296279237238997664168034971412265279319076363261368141455163766565
59839788489381733082668779901962886932296597379951931621187215455287394170243669885593888793316744533363119541518404088283815193421234122820
03095031334105070476015998798547252919066522247931971544033179483683737322082188577334162385644138070054191353024594391350255453188645479625
22602517629283743304651023610575835145507394433396102162296754614157811271970017386114942795014112532806212547758105129720884652631580948066
33687670147310733540717710876615935856814098212967730759197382973441445256688770855324570888958320993823432102718224114763732791357568615421
25284965790333509315277692550584564401055219264450531207375628774499816364633283581614033017581396735942732769044892036188038675495575180689
00585329272014939235005258451467069826285482578832673987352204572282392902071448222198855871028969919358730742778151597576207640239512438602
02032596596250212578349957710085626386118233813318509014686577064010676278617583772772895892746039403930337271873850536912957126715066896688
49388088514294360996201296675907922508227531381284985152690293170026313632894209579757795932763553116206675348865131732387243874806351331451
26448899675898288129254800764251865864902411111273013571971813816025831785069322440079986566353715440884548663931817083957357807990597308390
94881804060935959190907473960904410150516321749681412100765719177483767355751000733616922386537429079457803200042337452807566153042929014495
78062963413838355178359976470885134900485697369796523869584599459559209070905895689145114141268450546211794502661175016692826025095077077821
19504326173832235624376017767993627960993689751913949650333585071554184364568526166742436889203710374953284259271316105378349807407391586338
17967658425258036737206469351248652238481341663808061505704829059890696451936440018597120425723007316410009916987524260377362177763430621616
74488493081092990100951797454156425120482208671458684925513244426677712786372821133153622430109182439124338021404624222334915355951689081628
84879899882736304453724321742802157557779670216663170479697281724833928410156422745072717792693999297403080727703950135815451424940490265361
05825409373114653104943382484379718606937214444600826798002471229489405761853892203425608302697052876621377373594394224114707074072902725461
30735854174569141944648762435768239706570318416846754073346634629367398362000404140071405427763248013274220268539369886978760700959004868465
06267713630709798210065572851013066010107806337433447730734786538817426812307437660666433127753564665786037151929227684404582732832438082128
41218776132042460464900801054731426749260826922155637405486241717031027919996942645620955619816454547662045022411449404749349832206807191352
76798674781345820385957041346617793722853494003163159954409368408957253343870298671782977037333280680176463950209002394193149911500910527682
11195109990631661503115855828355826071794100525285836113699613034427901738117874120612881820620232638498615156564512300477929675636183457681
05043341769543067538041113928553792529241347339481050532025708728186307291158911335942014761872664291564036371927602306283840650425441742335
46454998705531872688792642410214736369862546374715974435494344389973005174252511087735788639094681209667342815258591992485764048805507132981
42993599114632399191139599267525763590074465728101918058418073422277347213977232182317717169164001088261125490933611867805757223910181861685
49108500885272274374212086524852372456248697662245384819298671129452945515497030585919307198497105414181636968976131126744027009648667545934
56705993699546450055892162804797636568613331656390739570327203438917541526750091501119885687270884819553167693168127289214303137681801644547
73675183534978579242764633541624336011259602521095016122641103460834656482355979342740568688849224458745493776752120324703803035491157544831 2
95275891939893680876327685438769557694881422844311998595700727521393176837831770339130423060958999137314684569010422095161967070506420256733
87344611565527617599272715187766001023894476053978951694570880272873622512107622409181006670088347473760515628553394356584375627124124445765
16630640859395079475509204639322452025354636344447917556617259621871992791865754908578529500128402290350615149373101070094461510116137124237
61426722541732055959202782129325725947146417224977321316381845326555279604270541871496236585252458648933254145062642337885651464670604298564
78196846159366328895429978072254226479040061601975197500746054515006029180663827149701611098795133663377137843441619405312144529185518013657
55586676150193730296919320761200092550650815832755084993407687972523699870235679310268041367457189566414318526790547171699629903630155456450
90044802789055701968328313630718997699153166679208958768572290600915472919636381673596673959975710326015571920237348580521128117458610065152
598883843114511894880552129145775699146577530041384717124577965048175856395072895337539755822087777506072339445587895905719156733