Níže uvedené úlohy představují přehled otázek, které se vyskytly v tomto nebo v minulých semestrech ve cvičení nebo v minulých semestrech u zkoušky. Mezi otázkami semestrovými a zkouškovými není žádný rozdíl, předpokládáme, že připravený posluchač dokáže zdárně zodpovědět většinu z nich.

Tento dokument je k dispozici ve variantě převážně s řešením a bez řešení.

Je to pracovní dokument a nebyl soustavně redigován, tým ALG neručí za překlepy a jazykové prohřešky, většina odpovědí a řešení je ale pravděpodobně správně :-).

Nahrazení rekurzivního volání tabelací

1.

Popište, jak byste výpočet hodnoty rekurzivní funkce f(6,7) převedli na vyplňování hodnot v poli, když funkce f je rekurzivně definována takto:

a)

$$\mathbf{f(x,y)} = \begin{cases} \mathbf{0} & \text{pokud x=0 nebo y=0 nebo z=0} \\ \mathbf{f(x-1, y-1)+ f(x-1,y) + f(x,y-1) +1} & \text{jinak} \end{cases}$$

b)

$$\mathbf{f(x,y)} = \begin{cases} \mathbf{0} & \text{pokud x=0 nebo y=0} \\ \mathbf{max\ (f(x-1, y-1)+ f(x-1,y)) + f(x,y-1) +1} & \text{jinak} \end{cases}$$

2.

Popište, jak byste výpočet hodnoty rekurzivní funkce f(6,8,7) převedli na vyplňování hodnot v poli, když funkce j je rekurzivně definována takto:

a)

$$\mathbf{f(x,y,z)} = \begin{cases} \mathbf{0} & \text{pokud x=0 nebo y=0 nebo z=0} \\ \mathbf{f(x-1, y-1, z-1)+ f(x-1,y,z) + f(x,y-1,z) + f(x,y,z-1)+1} & \text{jinak} \end{cases}$$

b)

$$\mathbf{f(x,y)} = \begin{cases} \mathbf{0} & \text{pokud x=0 nebo y=0} \\ \mathbf{3*f(x-1,y-1) - f(x,y-1) - f(x-1,y,z-1) + 1;} & \end{cases}$$

3.

Pascalův trojúhelník:

```
              1
            1   1
          1   2   1
        1   3   3   1
      1   4   6   4   1
    1   5  10  10   5   1
  1   6  15  20  15   6   1
1   7  21  35  35  21   7   1
.. .. .. .. .. .. .. .. .. .. ..
```

Pascalův trojúhelník obsahuje kombinační čísla nebo chcete-li binomické koeficienty. Označme binomický koeficient symbolem Bin(n,k).

Připomeňme si, že platí rekurentní vztah

Bin(n,k) = Bin(n-1,k)+Bin(n-1,k-1)

Můžeme proto rekurzivně definovat funkci Bin, která vrací binomický koeficient

$$\mathbf{Bin(n,k)} = \begin{cases} \mathbf{1} & \text{pro } n = 0 \text{ nebo } k = 0 \\ \mathbf{Bin(n-1,k)+Bin(n-1,k-1)} & \text{jinak} \end{cases}$$

Zjistěte kolikrát by byla funkce Bin() volána při provedení příkazu x = Bin(6,4);.

4.
Ackermanova funkce je definována pro dva celočíselné nezáporné parametry n, m, je tedy možné její hodnoty jednoduše zapisovat do tabulky.
Popište, jak budete postupně vyplňovat tabulku, abyste se vyhnuli rekurzivnímu volání.
Dokážete určit hodnotu A(4,4)?

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

(Chyba v definici není, druhý parametr ve třetím řádku je opět výsledek volání funkce A.)

Optimální binární vyhledávací strom

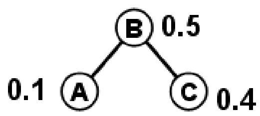
5.
Optimální binární vyhledávací strom
a) minimalizuje hloubku stromu
b) maximalizuje cenu uzlů
c) maximalizuje počet listů
d) minimalizuje dobu vyhledávání ve stromu
e) minimalizuje délku cesty z kořene do libovolného listu

6.
Je dáno n klíčů. Společně s každým klíčem je dána také pravděpodobnost dotazu na tento klíč. Nalezení kořene optimálního BVS sestaveného z těchto klíčů má asymptotickou složitost

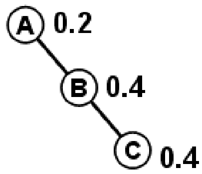
a) $O(\log(n))$
b) $\Theta(n)$
c) $O(n\cdot\log(n))$
d) $\Omega(n^2)$
e) $\Omega(2^n)$

7a.
Pravděpodobnost dotazu na jednotlivé konkrétní klíče BVS daného na obrázku je uvedena u jednotlivých uzlů. Předpokládejme, že se vždy dotazujeme na klíč, který ve stromu je. Z dlouhodobého hlediska je průměrný počet navštívených uzlů při jednom dotazu (= „cena stromu“) roven

a) 0.5
b) 1.0
c) 1.25
d) 1.5
e) 1.75



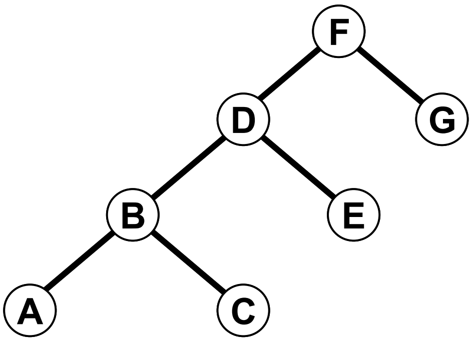
7b.
Pravděpodobnost dotazu na jednotlivé konkrétní klíče BVS daného na obrázku je uvedena u jednotlivých uzlů. Předpokládejme, že se vždy dotazujeme na klíč, který ve stromu je. Z dlouhodobého hlediska je průměrný počet navštívených uzlů při jednom dotazu (= „cena stromu“) roven

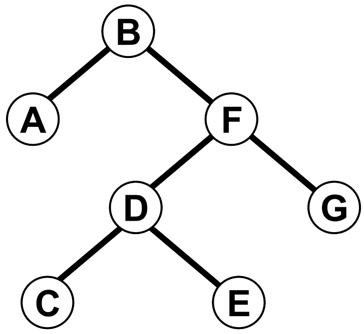


a) 0.2
b) 1.0
c) 1.8
d) 2.2
e) 2.5

8.
Jsou dány dva binární vyhledávací stromy obsahující stejné klíče. Pravděpodobnost dotazu na jednotlivé konkrétní klíče je dána níže uvedenou tabulkou. Vypočtěte, který z uvedených stromů je za těchto okolností výhodnější pro operaci FIND, tj, najděte ten, v němž průměrný počet dotazů na hodnotu klíče během jedné operace FIND je menší. (Předpokládáme, že obsah stromů se dlouhodobě nemění.)

A: 0.10
B: 0.20
C: 0.25
D: 0.05
E: 0.10
F: 0.25
G: 0.05





Řešení: Cenu každého uzlu vynásobíme jeho hloubkou (kořen má v tomto případě hloubku 1) a sečteme v každém stromě zvlášť:
Levý strom: 4*0.1 + 3*0.20 + 4*0.25 + 2*0.05 + 3*0.10 + 1*0.25 + 2*0.05 = 2.75
Pravý strom: 2*0.1 + 1*0.20 + 4*0.25 + 3*0.05 + 4*0.10 + 2*0.25 + 3*0.05 = 2.60
Výhodnější je pravý strom.

9.
Je možné, aby v optimálním BVS měl každý uzel nejvýše jednoho potomka?
Jaké by musely být pravděpodobnosti pro jednotlivé uzly v takovém případě?

Řešení Pokud má strom jen jeden nebo dva uzly, je požadavek splněn automaticky a pravděpodobnosti mohou být libovolné. Ukážeme si úvahu pro 3 uzly. Protože hledaný strom je vyhledávací a jeho kořen má jen jednoho potomka, musí být kořenem uzel s nejmenším nebo největším kíčem. Předpokládejme BÚNO, že je to uzel s nejnižším klíčem. Označme uzly v rostoucím pořadí klíčů u1, u2, u3 a pravěpodobnosti jejich návštěv p1, p2, p3. Uvědomme si nyní (kreslete si obrázek), že když má každý uzel nejvýše jednoho potomka, leží každý uzel v jiné hloubce. Předpokládejme, opět bez velké újmy na obecnosti, že uzel pi leží v hloubce i. Celkově tedy můžeme cenu našeho stromu vyjádřit formulí p1*1 + p2*2 + p3*3. Protože je to strom optimální, musí platit nerovnosti vyjadřující skutečnost, že tento strom nemá vyšší cenu než ostatní možné stromy nad těmito uzly. Jmenovitě máme:
p1*1 + p2*2 + p3*3 ≤ 2 *p1 + p2 + 2 * p3
p1*1 + p2*2 + p3*3 ≤   3 *p1 + 2 * p2 + p3
Po jednoduché úpravě získáme podmínku pro pravděpodobnosti:

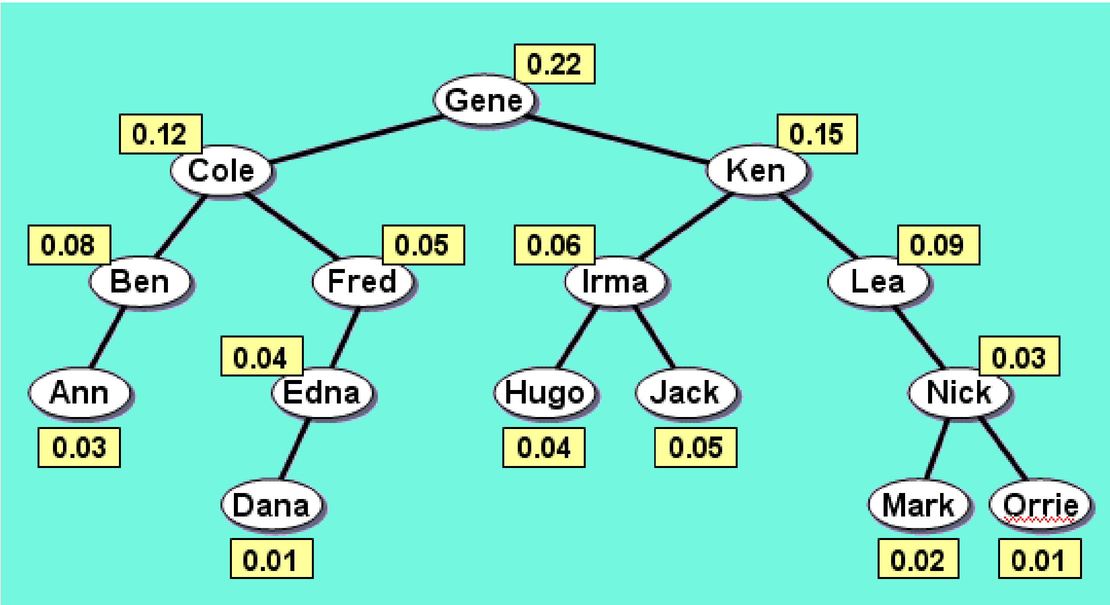
$p2 + p3 \leq p1$.
Pro tři uzly tak například vyhovují pravděpodobnosti p1 = 0.6, p2 = 0.3, p3 = 0.1 apod.
Ukažte, nebo alespoň ověřte na příkladu, že pro čtyři uzly v podobné konfiguraci musí platit

$p3 + p4 \leq p2$,

$p2 + p3 + p4 \leq p1$.

10.

Nejprve připomeneme příklad z přednášky:



Kořeny optimálních podstromů jsou popsány polem

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

Ceny optimálních podstromů jsou popsány polem:

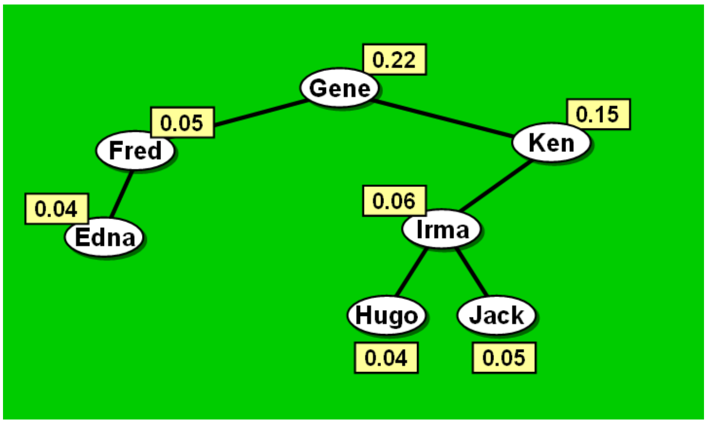
	1-A	2-B	3-C	4-D	5-E	6-F	7-G	8-H	9-I	10-J	11-K	12-L	13-M	14-N	15-O
1-A	0.03	0.14	0.37	0.39	0.48	0.63	1.17	1.26	1.42	1.57	2.02	2.29	2.37	2.51	2.56
2-B	0	0.08	0.28	0.30	0.39	0.54	1.06	1.14	1.30	1.45	1.90	2.17	2.25	2.39	2.44
3-C	0	0	0.12	0.14	0.23	0.38	0.82	0.90	1.06	1.21	1.66	1.93	2.01	2.15	2.20
4-D	0	0	0	0.01	0.06	0.16	0.48	0.56	0.72	0.87	1.32	1.59	1.67	1.81	1.86
5-E	0	0	0	0	0.04	0.13	0.44	0.52	0.68	0.83	1.28	1.55	1.63	1.77	1.82
6-F	0	0	0	0	0	0.05	0.32	0.40	0.56	0.71	1.16	1.43	1.51	1.63	1.67
7-G	0	0	0	0	0	0	0.22	0.30	0.46	0.61	1.06	1.31	1.37	1.48	1.52
8-H	0	0	0	0	0	0	0	0.04	0.14	0.24	0.54	0.72	0.78	0.89	0.93
9-I	0	0	0	0	0	0	0	0	0.06	0.16	0.42	0.60	0.66	0.77	0.81
10-J	0	0	0	0	0	0	0	0	0	0.05	0.25	0.43	0.49	0.60	0.64
11-K	0	0	0	0	0	0	0	0	0	0	0.15	0.33	0.39	0.50	0.54
12-L	0	0	0	0	0	0	0	0	0	0	0	0.09	0.13	0.21	0.24
13-M	0	0	0	0	0	0	0	0	0	0	0	0	0.02	0.07	0.09
14-N	0	0	0	0	0	0	0	0	0	0	0	0	0	0.03	0.05
15-O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.01

Analogicky podle příkladu z přednášky určete, jak bude vypadat optimální strom, když jej vybudujeme pouze pro 7 uzlů počínaje Ednou a konče Kenem.

Řešení:

	5-E	6-F	7-G	8-H	9-I	10-J	11-K
0	0.04	0.13	0.44	0.52	0.68	0.83	1.28
0	0	0.05	0.32	0.40	0.56	0.71	1.16
0	0	0	0.22	0.30	0.46	0.61	1.06
0	0	0	0	0.04	0.14	0.24	0.54
0	0	0	0	0	0.06	0.16	0.42
0	0	0	0	0	0	0.05	0.25
0	0	0	0	0	0	0	0.15
0	0	0	0	0	0	0	0

	5-E	6-F	7-G	8-H	9-I	10-J	11-K
0	5	6	7	7	7	7	7
0	0	6	7	7	7	7	7
0	0	0	7	7	7	7	7
0	0	0	0	8	9	9	11
0	0	0	0	0	9	9	11
0	0	0	0	0	0	10	11
0	0	0	0	0	0	0	11
0	0	0	0	0	0	0	0



11.

Jsou dány prvky s klíči A-G. Pravděpodobnost dotazu na jednotlivé konkrétní klíče je dána níže uvedenou tabulkou. Metodou dynamického programování sestrojte optimální strom z hlediska operace FIND, tj, najděte takový strom, v němž průměrný počet dotazů na hodnotu klíče během jedné operace FIND je nejmenší (Předpokládáme, že obsah stromu se dlouhodobě nemění). Napište, jak vypadá tabulka ohodnocení optimálních podstromů a tabulka jejich kořenů a výsledný strom namalujte.

A: 0.10
B: 0.10
C: 0.25
D: 0.35
E: 0.10
F: 0.05
G: 0.05

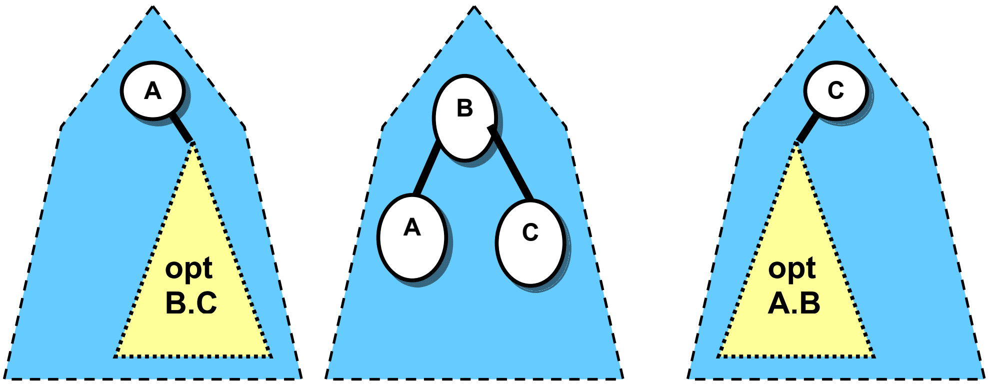
Řešení:

Nejlepší je se podívat do přednášky, kde je vše pěkně popsáno a definováno. Zde jen několik „praktických“ poznámek.

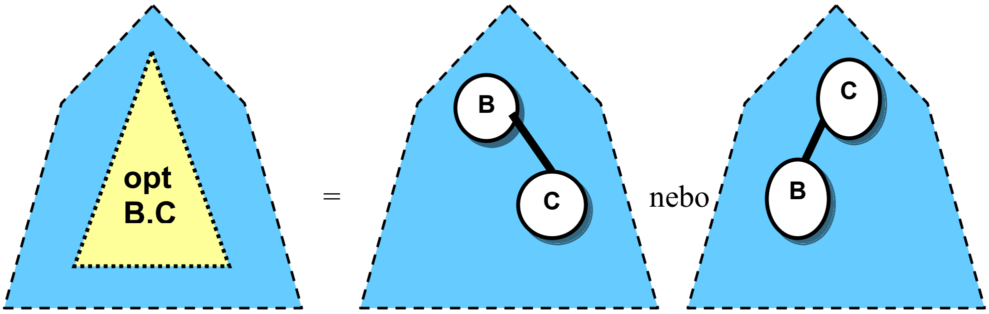
Tabulka ohodnocení optimálních podstromů *t*

- Má stejný počet řádek, jako je počet uzlů. Řádkový index odpovídá počátečnímu uzlu *L* z intervalu uzlů *L..R,*
- má o jeden sloupec více, než je uzlů. Sloupcový index odpovídá koncovému uzlu *R* z intervalu uzlů *L..R,.*
- je dobré si uvědomit, že uzly mají pevně dané pořadí (jejich klíče jsou uspořádány)
- má tedy pod diagonálou samé nuly, protože podstrom s L < R nemá smysl,
- vše pod diagonálou chápeme jako prázdný podstrom,
- na diagonále jsou hodnoty „jednouzlových“ podstromů, tedy přímo pravděpodobnost dotazu (1-krát, neboť hloubka je 1. V předchozích přednáškách byla hloubka stromu s jedním uzlem definována jako rovna nule. Uváděním hloubky od jedné vznikl trochu zmatek. Konzistentnější by bylo ponechat hloubku stromu s jedním uzlem rovnu nule a uvádět, že hodnotu pravděpodobnosti dotazu uzlu vynásobíme hloubkou uzlu zvětšenou o 1.),

- Při vyplňování tabulky musíme do políčka o souřadnicích *t*[L,R] vyzkoušet všechny možné polohy kořene v rámci intervalu L..R. Pro L=A a L = C jsou možnosti následující:



Opt B,C znamená optimální podstrom sestávající z uzlů B a C, tj. výsledek předchozího kroku dynamického programování. Je to ten, jehož ohodnocení bylo menší, v našem případě podstrom s kořenem C.

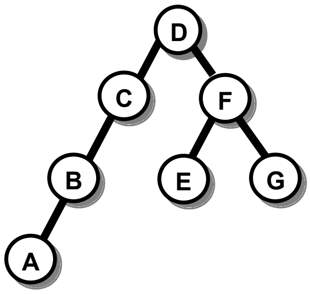


		A	B	C	D	E	F	G
A	0	0.1	0.3	0.75	1.45	1.75	1.9	2.1
B	0	0	0.1	0.45	1.15	2.03	1.5	1.7
C	0	0		0.25	0.85	1.05	1.2	1.4
D	0	0			0.35	0.55	0.7	0.9
E	0	0				0.1	0.2	0.35
F	0	0					0.05	0.15
G	0	0	0					0.05

		A	B	C	D	E	F	G
A	-	A	AB	C	C	CD	D	D
B	-	-	B	C	CD	D	D	D
C	-	-		C	D	D	D	D
D	-	-			D	D	D	D
E	-	-				E	E	EFG
F	-	-					F	FG
G	-	-						G

Zažlucená políčka jsou nejednoznačná, protože vyjde stejná hodnota pro více různých kořenů.

Např. hodnota pro podstrom BCD na pozici t[B,D] se vypočítá jako:
0.1+0.25+0.35 + min(0+0.85, 0.1+0.35, 0.45+0) = 0.7 + min(0.85, 0.45, 0.45) = 1.15
Výsledný strom tedy může vypadat například takto:



11bc.
Zopakujte předchozí úlohu pro uzly s pravděpodobnostmi:

b)	c)
A: 0.15 B: 0.15 C: 0.20 D: 0.25 E: 0.10 F: 0.10 G: 0.05	A: 0.10 B: 0.10 C: 0.25 D: 0.25 E: 0.10 F: 0.15 G: 0.05

Uvádíme tabulky cen a kořenů, strom rekonstruujte jako cvičení sami.
b)

0	0.15	0.45	0.85	1.45	1.75	2.10	2.25
0	0	0.15	0.50	1.00	1.30	1.60	1.75
0	0	0	0.20	0.65	0.85	1.15	1.30
0	0	0	0	0.25	0.45	0.75	0.90
0	0	0	0	0	0.10	0.30	0.40
0	0	0	0	0	0	0.10	0.20
0	0	0	0	0	0	0	0.05

0	5	5	6	7	7	8	8
0	0	6	7	7	7	8	8
0	0	0	7	8	8	8	8
0	0	0	0	8	8	8	8
0	0	0	0	0	9	9	10
0	0	0	0	0	0	10	10
0	0	0	0	0	0	0	11

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

c)

0	0.10	0.30	0.75	1.25	1.55	2.05	2.20	0	5	5	7	7	7	8	8
0	0	0.10	0.45	0.95	1.25	1.65	1.80	0	0	6	7	7	7	8	8
0	0	0	0.25	0.75	0.95	1.35	1.50	0	0	0	7	7	8	8	8
0	0	0	0	0.25	0.45	0.85	1.00	0	0	0	0	8	8	8	8
0	0	0	0	0	0.10	0.35	0.45	0	0	0	0	0	9	10	10
0	0	0	0	0	0	0.15	0.25	0	0	0	0	0	0	10	10
0	0	0	0	0	0	0	0.05	0	0	0	0	0	0	0	11
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Nejdelší společná podposloupnost

11.
Dva dané řetězce mají oba délku n. Nejdelší společnou podposlounost těchto řetězců lze nalézt v čase

a) $\Theta(\log(n))$
b) $\Theta(n)$
c) $\Theta(n \cdot \log(n))$
d) $\Theta(n^2)$
e) $\Theta(n^3)$

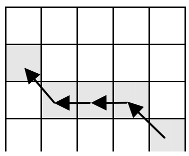
12.
Nejdelší společná podposloupnost
Algoritmus hledání nejdelší společné posloupnosti je snadno zapamatovatelný a mechanicky reprodukovatelný.
Najděte tedy nejdelší společnou podposloupnost v několika jednoduchých případech:
A: 1100110011001100
B: 1010101010101010

A: 1101001000100001000001000001
B: 0010110111011110111110111110 (B = doplněk A)

A: 1101001000100001000001000001
B: 1000001000001000010001001011 (B = A pozpátku)

13a.
Následující matice 5x4 polí reprezentuje výřez v obrázku. Každé políčko představuje jeden pixel s uvedenou hodnotou. Postupovat lze jen zleva doprava, a to vodorovně, nebo pod úhlem ±45 stupňů, tj. do políčka s indexem [i,j] se lze dostat z políčka v předchozím sloupci s indexem [i-1, j-1], [i, j-1], [i+1, j-1], kde i=řádkový index a j= sloupcový index.

$$\begin{array}{c|cccc} & j \rightarrow & & & \\ \hline i & 2 & 12 & 23 & 18 \\ | & 23 & 23 & 6 & 12 \\ | & 20 & 9 & 12 & 10 \\ v & 18 & 15 & 11 & 8 \\ & 16 & 12 & 8 & 7 \end{array}$$



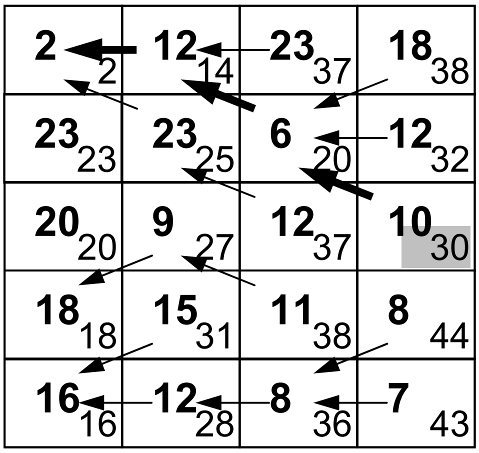
Dynamickým programováním nalezněte spojnici levé a pravé strany (touto spojnicí se míní posloupnost políček, jedno políčko na sloupec, začínající ve sloupci s indexem j=1 a končící ve sloupci s j=4) takovou, že součet hodnot podél ní je minimální.

Jaké mezivýsledky si musíte pamatovat pro jednotlivé sloupce a pro celou tabulku?

Myšlenka řešení:
Představme si, že máme matici M obsahující pouze dva sloupce. Abychom našli uvedenou spojnici, uděláme toto: Probereme postupně všechny prvky ve druhém sloupci a pro každý prvek M[i,2] zaregistrujeme minumum ze tří součtů: S[i,2] = min {M[i-1,1]+M[i,2], M[i,1]+M[i,2], M[i+1,1]+M[i,2]}, tedy součtu hodnoty M[i,2] a některého z jeho možných tří předchůdců. Pro toto minimum zaregistrujeme také, pro kterého z předchůdců nastává (např, pomocí hodnot –1, 0, +1).
Když toto provedeme, stačí jen projít znova druhý sloupec a najít nejmenší hodnotu součtů S[i,2]. Protože je tam zaregistrován i přechůdce, je kompletní (byť krátká) hledaná cesta nyní známa. Podívejte se na následující obrázek. Velká čísla představují původní hodnoty M[i,j], malá čísla představují součty podél nalezených spojnic končících v jednotlivých prvcích druhého sloupce, tj hodnoty S[i,j] (pro zjednodušení případné implementace položme S[i,1] = M[i,1] pro všechna i). Šedě je podbarveno celkové minimum.

2 2	**12** 14
23 23	**23** 25
20 20	**9** 27
18 18	**15** 31
16 16	**12** 28

Nyní si představme, že k tomuto řešení připojíme dále třetí sloupec. Ve druhém sloupci je pro každý jeho prvek známa optimální spojnice, která končí v tomto prvku. Nyní tedy stačí projít třetí sloupec stejně jako jsme poprvé procházeli druhý sloupec a zaregistrovat pokaždé minimum S[i,3] = min {S[i-1,2]+M[i,3], S[i,2]+M[i,3], S[i+1,2]+M[i,3]} a zaregistrovat příslušného předchůdce.
Přidávání dalších (čtvrtého, pátého, atd...) sloupců se řídí stejným pravidlem, takže v dané úloze získáváme následující definitivní obrázek



Celkem tedy si pro každý sloupec j musíme pamatovat hodnoty S[i,j] i=1..5, a předchůdce každého prvku. Předchůdce můžeme ukládat do další matice P, kde P[i,j] je –1, 0 nebo +1.

13b.

Následující matice *m* reprezentuje výřez v obrázku. Každé políčko představuje jeden pixel s uvedenou hodnotou. Postupovat lze jen zleva doprava, a to vodorovně, nebo pod úhlem ±45 stupňů, tj. do políčka s indexem [i,j] se lze dostat z políčka v předchozím sloupci s indexem [i-1, j-1], [i, j-1], [i+1, j-1], kde i=řádkový index a j= sloupcový index.

j →

i				
i	3	12	23	18
|	15	23	6	12
|	20	9	12	10
v	18	14	19	8
	16	12	8	7

Dynamickým programováním nalezněte spojnici levé a pravé strany (touto spojnicí se míní posloupnost políček, jedno políčko na sloupec, začínající ve sloupci s indexem j=1 a končící ve sloupci s j=4) takovou, že součet absolutních rozdílů hodnot podél cesty je minimální. Rozdílem hodnot podél cesty je např. pro políčko [i, j] hodnota abs(m[i,j] – m[i_{prev},j-1]), kde i_{prev} nabývá jedné z hodnot {i-1, i, i+1}.

Jaké mezivýsledky si musíte pamatovat pro jednotlivé sloupce a pro celou tabulku?

Myšlenka řešení je zcela identická jako v předchozí variantě. Místo pouhých součtů podél spojnice budeme registrovat součty absolutních hodnot rozdílů, jinak se nezmění vůbec nic.
Matici S tedy budeme počítat ze vztahu:
S[i,j] = min {S[i–1, j–1] + abs(M[i, j] – M[i–1, j–1]),
S[i, j–1] + abs(M[i, j] – M[i, j–1]),
S[i+1, j–1] + abs(M[i, j] – M[i+1, j–1]) }

Výslednou matici tu pro nedostatek času neuvádím, spočtěte si ji jako snadné cvičení.

-- **E** --

A

The set of n distinct keys is given. Also a probability of the key being queried is given for each key. Establishing the root of the optimal BST built on the given keys is a task with complexity:

f) $O(\log(n))$
g) $\Theta(n)$
h) $O(n \cdot \log(n))$
i) $\Omega(n^2)$
j) $\Omega(2^n)$

A elsewhere

Two given strings are both of length n. The longest comon subsequence of these strings can be found in time:

f) $\Theta(\log(n))$
g) $\Theta(n)$
h) $\Theta(n \cdot \log(n))$
i) $\Theta(n^2)$
j) $\Theta(n^3)$