Servisní stanice - komentář a řešení

Označme OC(k, m) cenu za optimální rozmístění k stanic a údržbu úseků mezi nimi na prvních m kilometrech trati s tím, že poslední k-tá stanice stojí právě na m-tém kilometru. Dále cenu za údržbu úseku délky d kilometrů označme CUU(d).

Řešení celé úlohy získáme tak, že poslední N-tou stanici zkusíme umístit na všechny možné kilometry, t.j. necháme parametr m probíhat od N až do L−1 a pokaždé k hodnotě OC(N, m) přičteme cenu za údržbu úseku z N-té stanice do cíle. Ze všech těchto součtů OC(N, m)+CUU(L−m) vybereme minimální. V řeči symbolů:

Řešení = min { OC(N, m)+CUU(L−m); m = N ... L−1}.

Zbývá sestavit algortimus pro výpočet hodnot OC(k, m).

Když stojí k-tá stanice na m-tém kilometru, pak těsně předchozí (k−1) stanice může stát kdekoli na celých kilometrech počínaje (k−1) kilometrem a konče (m−1) kilometrem. Ze všech těchto možností je nutno vybrat optimální, to jest nalézt nejlepší polohu pro stanici (k−1). Předpokládejme, že umíme spočítat optimální rozložení (k−1) stanic na každé počáteční části tratě v délce (k−1) až (m−1) kilometrů. To znamená, že umíme spočítat hodnoty OC(k−1, k−1) až OC(k−1, m−1). K ceně za optimální umístění (k−1) stanic je navíc nutno přičíst cenu za údržbu úseku z (k−1) do k-té stanice a cenu za výstavbu stanice na m-tém kilometru. Celkem tedy získáváme rekurentní vztah

$OC(k, m) = \min \{ OC(k-1, m1)+CUU(m-m1)+s_m;\ m1 = k-1 \ldots m-1\}$ pro $1 < k \leq m < L$.

Pokud stavíme první stanici, tj. k = 1, pak cena je přímo rovna pouze nákladům na provoz prvního úseku a výstavbu první stanice:

$OC(1, m) = CUU(m)+s_m$, pro $1 \leq m < L$.

Hodnoty s_m jsou dány, hodnoty CUU(m) lze snadno počítat nebo předpočítat do tabulky a hodnoty OC(k, m) lze ukládat do 2D tabulky a vyplnit ji podle uvedených vztahů. Nakonec, v posledním cyklu podle prvního uvedeného vztahu spočteme hledané řešení úlohy.

Jak ukazuje přiložený výpis, kód řešení se vejde na jednu stránku. Lze v něm dále provádět drobné lokální optimalizace, např. přičítání hodnoty s[m] lze vyjmout z cyklu, tabulku OC není třeba vyplňovat úplně celou (hodnoty OC(k,m) pro m > k+L−N−1 se ve výpočtu řešení nevyužijí), apod .

```
static int N, L;      // no of stations and track length
static int [] s;      // s[i] cost of station at i-th km
static int [][] OC;   // DP table NxL
static int a, b;      // params of segment quadratic cost
static int [] CUU;    // CUU[i] cost of maintenance of segment length i km

static int calcResult(){
  // segment from 1st station to 0th station, incl cost of 1st station
  for(int m = 1; m <= L-N; m++)
    OC[1][m] = CUU[m]+ s[m];

  // segment from k-th station to (k-1)-th station, incl cost of k-th station
  int currVal, minVal;
  for(int k = 2; k <= N; k++)
    for(int m = k; m < L; m++) {
      minVal = Integer.MAX_VALUE;
      for(int m1 = k-1; m1 < m; m1++)  {
        currVal = OC[k-1][m1] + CUU[m-m1] + s[m];
        if (currVal < minVal )
          minVal = currVal;
      }
      OC[k][m] = minVal ;
    }

  // segment from end of track to N-th station
  minVal = Integer.MAX_VALUE;
  for(int m = N; m < L; m++) {
    currVal = OC[N][m] + CUU[L - m];
    if (currVal < minVal)
      minVal = currVal;
  }
  return minVal;
}

static void readAndInitAll() throws IOException {
  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
  StringTokenizer st = new StringTokenizer(br.readLine());
  L = Integer.valueOf(st.nextToken());
  N = Integer.valueOf(st.nextToken());
  OC = new int [N+1][L+1];  // all indices start at 1;
  st = new StringTokenizer(br.readLine());
  a = Integer.valueOf(st.nextToken());
  b = Integer.valueOf(st.nextToken());
  CUU = new int [L];
  for(int i = 1; i < L; i++)
    CUU[i] = i*(a*i+b);
  s = new int [L];
  st = new StringTokenizer(br.readLine());
  for(int i = 1; i < L; i++)
    s[i] = Integer.valueOf(st.nextToken());
}

public static void main(String[] args) throws IOException  {
  readAndInitAll();
  System.out.printf("%d\n",calcResult());
}
```