АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА
РЕДУКЦИЯ

Дефиницията на операцията reduce(i) може да бъде преформулирана по следния начин: Върху числовата ос избираме n точки, съответстващи на елементите на числовата редица. Получават се n-1 отсечки, свързващи двойките съседни точки. В началото всички n-1 отсечки се изтриват, така че всяка точка остава самостоятелна. Еднократното прилагане на reduce() е еквивалентно на построяването на една от тези отсечки, което води до обединяване на точките, съответстващи на краищата на тази отсечка. Тази интерпретация на операцията reduce() опростява обясненията на описаните по-долу решения (тъй като в тази версия последователността никога не се скъсява). Стойността на тази операция е равна на по-големия елемент на участващите два края на отсечката (двата съседни елемента a_i и a_{i+1} на редицата).

1. $O(n^3)$ решение – динамично програмиране

Подредица или сегмент от числовата редица ще наричаме всяка нейна непрекъсната последователност: a_l, a_{l+1},..., a_r. В най-простото решение за всеки сегмент на редицата ще изчислим минималната цена за редуцирането му до един елемент (за сегмента a_l,.a_{l+1} ,..., a_r означаваме тази стойност с tab[l, r]). Очевидно е, че за всяко $i \in \{1,..., n\}$, tab[i, i] = 0. Ако $l < r$, последната операция reduce() от процеса на редуциране на този сегмент до един елемент винаги ще струва $M = \max(a_l,...,a_r)$. В предлаганото решение можем само да избираме позицията i за прилагане на reduce(), което на практика е позицията на линейния сегмент, създаден по време на тази операция reduce(). Ето защо за $l < r$ е валидна формулата:

`tab[l, r] = min(tab[l, i] + tab[i+1, r] + M)`, където $l \leq i \leq r$.

Използвайки тази формула, можем да изчислим всички стойности на масива tab - от най-късите до най-дългите сегменти. Общата времева сложност на това решение е $O(n^3)$, тъй като размерността на масива tab е n^2, а за изчисляване на всеки елемент от масива tab изисква време O(n). Това решение дава ~ 30 от 100 възможни точки.

Примерна реализация:

```
/* O(n^3) dynamic programming solution */
#include <cstdio>
#include <algorithm>
using namespace std;
#define MAX_N 2000
#define INFTY 1000000000000000LL
int n, a[MAX_N];
long long tab[MAX_N][MAX_N]; /* for DP */
int main()
{
  scanf("%d", &n);
```

```
  for (int i = 0; i < n; i++)
    scanf("%d", &a[i]);
  /* tab[l][l] = 0 is automatic */
  for (int len = 2; len <= n; len++)
    for (int l = 0; l < n; l++)
    {
      int r = l + len - 1;
      if (r >= n) break;
      /* counting tab[i][j] */
      int mx = 0;
      for (int i = l; i <= r; i++)
        mx = max(mx, a[i]);
      tab[l][r] = INFTY;
      for (int i = l; i < r; i++)
        tab[l][r] = min(tab[l][r], tab[l][i] + tab[i+1][r] +
mx);
    }
  printf("%lld\n", tab[0][n - 1]);
  return 0;
}
```

2. Търсене на Greedy решение

Да се опитаме да постигнем по-бързо Greedy решение. Да допуснем, че $a_0 = a_{n+1} = +\infty$. Отляво и от дясно заграждаме дадената числова редица с много големи елементи. Твърдим, че n - 1 кратното изпълнение на следните стъпки:

1. Намираме такова i, за което: $a_{i-1} \leq a_i \leq a_{i+1}$
2. Ако $a_{i-1} < a_{i+1}$, то изпълни reduce(i − 1), в противен случай изпълни reduce(i),

ни гарантира оптимална схема за редуциране на дадената редица.

Доказателство: На първо място трябва да докажем, че за всяка от n-1 стъпки, намирането на необходимата стойност на i е възможно. За всяка стъпка, нека започнем с j = 1 и докато условието $a_{j-1} > a_j$ е изпълнено увеличаваме стойността на j с единица. Този процес със сигурност ще се прекрати, тъй като $a_n < a_{n+1} = +\infty$ (докато $n \geq 1$). Стойността на i = j-1 в момента на прекратяване е индекса, който удовлетворява точка 1.

Доказателството, че тази схема за редукция е оптимална се извършва по метода на математическата индукция. За n = 1 това решение очевидно е оптимално с цена 0. Ако n> 1 трябва да докажем, че ако съществува оптимална схема за редукция на дадената редица за i=n, то схемата за редукция за i=n+1, също е оптимална. Това се постига с допускане на противното.

Автор Пано Панов