Розв’язання

Розглянемо найпростіший алгоритм розв’язання задачі. Перебиратимемо всі трійки відрізків.
Для кожної трійки треба перевірити можливість існування відповідного трикутника. Якщо трикутник існує, підрахуємо сумарну вартість трьох відрізків, а з усіх знайдених вартостей виберемо найменшу. Складність алгоритму — N(N3).
А от швидше розв’язання. Розглядатимемо всі відрізки в порядку від найкоротших до найдовших і підтримуватимемо таку структуру даних: масив F, у комірці F[C] якого зберігається найбільша сумарна довжина двох відрізків (з числа перших N відрізків) із загальною вартістю c. Оскільки ціни відрізків не перевищують N, сумарна вартість двох відрізків не буде більшою за 2N. Переходячи до чергового ((k + 1)-го) відрізка — нехай цей відрізок має довжину L та вартість C — ми шукаємо найменше таке значення c, для якого F[c] >L. Це можна зробити простим лінійним пошуком. Для знайденого значення c сума C + c буде найменшою можливою вартістю трикутника, найбільшою стороною якого є розглядуваний відрізок. Якщо ця сума менша за поточний мінімум знайдених вартостей, відповідним чином цей мінімум оновлюємо. Крім того, оновлюємо і сам масив F[c]. Для цього пробуємо взяти в пару з (k + 1)-м відрізком почергово всі відрізки від першого до k-го і оновлюємо, якщо потрібно, відповідну комірку масиву F. Таким чином, на кожному з N кроків ми виконуємо з N кроків ми виконуємо O(N) операцій, тож загальна складність алгоритму — $O(N^2)$.

Тепер розглянемо оптимальне розв’язання. Воно базується на такій нескладній властивості:
Властивість. Якщо серед натуральних чисел L1, L2, ..., Lk немає трьох таких, що утворюють сторони трикутника, то найбільше з цих чисел не може бути меншим за K-й член послідовності Фібоначчі. Послідовність Фібоначчі задається таким чином: $F_1 = F_2 = 1$, $F_k = F_{k-1} + F_{k-2}$, K>= 3.
Це твердження можна довести методом математичної індукції, попередньо впорядкувавши числа L1, L2, ..., Lk за неспаданням. Оскільки $F_{45} > 109$ (у чому можна переконатися, написавши спеціальну «дослідницьку» програму), серед будь-яких 45 натуральних чисел, які не перевищують 109, обов’язково знайдуться три, що є сторонами трикутника. Отже, зокрема і серед 45 найдешевших відрізків знайдуться три, що утворюють трикутник. Але ціни відрізків — перестановка чисел від 1 до N, тому сумарна ціна цих трьох відрізків не може перевищувати 45 + 44 + 43 = 132. Таким чином, усі відрізки вартістю більше за 132 можна відкинути й шукати потрібну трійку лише серед

тих, що залишилися. Маємо фактично лінійне від N розв’язання з додатком, що складає порядку 132^3 операцій.

Варіант 1

Варіант 2

```
#include "fstream"
```

```
using namespace std;
```

```
long int l[1000000],c[1000000],s,n,i,j,k;
```

```
ifstream cin("segments.dat");
```

```
ofstream cout("segments.sol");
```

```
int main()
```

```
{cin>>n;
```

```
for(i=0;i<n;i++)cin>>l[i]>>c[i];
```

```
s=2000000000;

for(i=0;i<n-2;i++)

for(j=i+1;j<n-1;j++)

for(k=j+1;k<n;k++)

if (l[i]<l[j]+l[k] && l[j]<l[i]+l[k] && l[k]<l[j]+l[i]  && c[i]+c[j]+c[k]<s)

s=c[i]+c[j]+c[k];

if (s<2000000000)cout<<s<<endl;else cout<<-1<<endl;

return 0;

}

#include "fstream"

using namespace std;

long int a[10000
```

```
ifstream cin( "segments.dat" );

ofstream cout( "segments.sol" );

int main()

{cin>>n;

k=3000;

for (i=0;i<n;i++){cin>>l>>c;

a[c]=l;}

ans=2000000000;

for (i = 1; i <= n; i++)

for (j = i + 1; j <= n; j++)

for (u = j + 1; u <= n; u++)

if (i + j + u < ans && a[i] < a[j] + a[u] && a[j] < a[i] + a[u] &
```

```
if (ans==2000000000)ans=-1;

cout<<ans<<endl;

return 0;

}
```