Nama : Fajar Satria

NRP : 05111940000083

Kelas : B

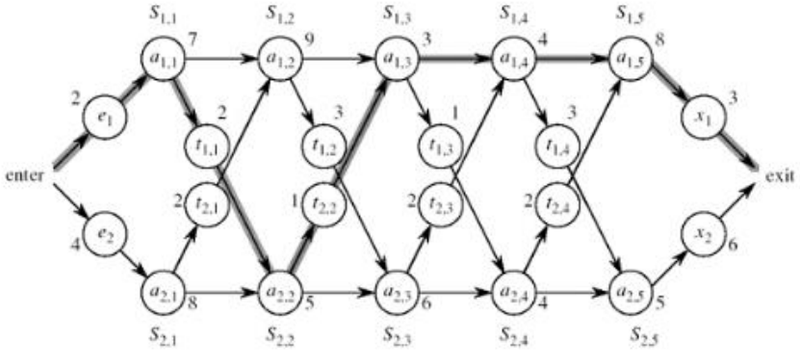
Resume Kuliah PAA – 10 Mei 2021

Algoritma Dynamic Programming dalam banyak kasus dapat diselesaikan dengan algortima yang bersifat top down dengan menggunakan memoisasi tabel. Namun, sebenarnya seluruh konsep programming itu dapat diselesaikan dengan metode bottom up. Meskipun akan merepotkan untuk mencari bagaimana agar permasalahan dapat diselesaikan dengan metode bottom up tersebut. Berikut adalah pembahasan soal yang berkaitan dengan metode bottom up tersebut.

a. Assembly Line

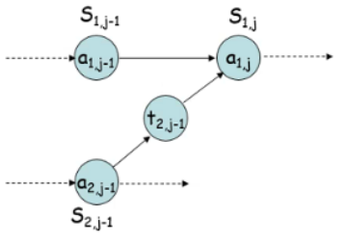
Rangkuman soal untuk permasalahan ini adalah sebagai berikut

- Terdapat dua jalur untuk mengerjakan tiap part dari mobil tersebut. Dimana barang yang akan dikerjakan dapat berpindah jalur.
- Tiap jalur memiliki n pos $S_{1,1}$, … , $S_{1,n}$ dan $S_{2,1}$, … , $S_{2,n}$
- Dari kedua jalur tersebut tiap pos $S_{1,j}$ dan $S_{2,j}$ mengerjakan part yang sama tetapi memakan waktu yang berbeda.
- Tiap perpindahan jalur memerlukan tambahan waktu $t_{1,j}$ dan $t_{2,j}$
- Ketika barang masuk ke pos pertama memerlukan tambahan waktu e_1 dan e_2
- Ketika barang sudah selesai dari pos terakhir memerlukan tambahan waktu x_1 dan x_2



Setelah memahami persoalan diatas, pastinya akan terpikirkan untuk membuat solusi bruteforce yaitu dengan mengiterasi semua kemungkinan yang ada untuk mendapatkan solusi yang benar. Akan tetapi hal tersebut akan memakan waktu yang tidak sedikit. Misalnya untuk pos yang sangat banyak, tentu akan memerlukan waktu yang sangat banyak pula.

Kembali lagi ke persoalan, solusi yang paling mungkin adalah mencari waktu tercepat untuk tiap pos $S_{1,j}$ atau $S_{2,j}$. Misalnya untuk $S_{1,2}$, akan ada 2 kemungkinan yaitu barang datang dari pos $S_{1,1}$ tanpa tambahan waktu atau barang datang dari pos $S_{2,1}$ dengan konsekuensi tambahan waktu $t_{2,1}$ seperti gambar disamping kanan. Dari kedua kemungkinan tersebut akan dipilih pos yang memiliki waktu tersepat.



Mekanisme perhitungan waktu tiap pos tersebut berlaku untuk **j=2,3,…,n**. Sedangkan untuk **j=1** cukup mencari waktu **e** ditambah waktu pada pos yang memakan waktu paling sedikit. Jadi dapat di misalkan sebagai berikut

f* = waktu tercepat untuk menyelesaikan pengerjaan seluruh part mobil.

$f_i[j]$ = waktu tercepat mulai dari awal sampai **$S_{i,j}$**.

Jadi **f*** dapat di hitung dengan

$f^* = \min(f_1[n] + x_1, f_2[n] + x_2)$

Dan **f[i][j]** dapat dihitung dengan

- untuk **j=1**

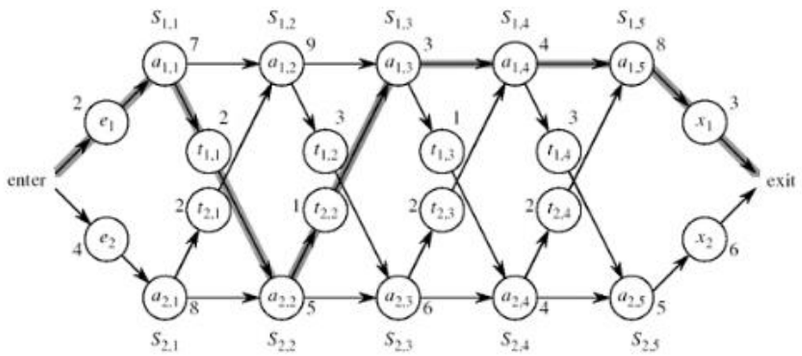
  $f_1[1] = e1 + a1,1$

  $f_2[1] = e2 + a2,1$

- untuk **j=2, 3, …., n**

  $f_1[j] = \min(f_1[j-1] + a_{1,j}, f_2[j-1] + t_{2,j-1} + a_{i,j})$

  $f_2[j] = \min(f_2[j-1] + a_{2,j}, f_1[j-1] + t_{1,j-1} + a_{i,j})$



Jadi dapat dibuatkan tabel untuk mencari waktu minimum dari tiap pos yang dilewati barang tersebut sebagai berikut.

j	1	2	3	4	5
f1[j]	9	$18^{[1]}$	$20^{[2]}$	$24^{[1]}$	$32^{[1]}$
f2[j]	12	$16^{[1]}$	$22^{[2]}$	$25^{[1]}$	$30^{[1]}$

Meskipun tabel diatas berukuran **2xN**, namun pada implementasinya dapat diperkecil menjadi tabel berukuran **2x2**. Hal ini dapat dilakukan karena relasi rekurensi pada permasalah soal ini adalah hanya antara **j** dan **j-1**. Misalnya untuk j=19, cukup berelasi dengan j=18, tidak akan menyentuh j=17 atau bahkan j=1. Sehingga dalam implementasinya cukup menggeser row dari tabel tersebut ketika sudah selesai dihitung.

PSEUDOCODE

```
MIN(a, b)
1. if a>b
2.         return b
3. else
4.         return a
SOLVE()
1. let dp[0..2][0..2] be a new 2D array
2. let a[0..2][0..1024] be a new 2D array
3. let t[0..2][0..1024] be a new 2D array
4. read n
5. read e1
6. read e2
7. for j=0 to 1
```

8. for i=0 to n-1
9. read a[j][i]
10. for j=0 to 1
11. for i=0 to n-2
12. read t[j][i]
13. read x1
14. read x2
15. dp[0][0] = e1 + a[0][0]
16. dp[1][0] = e2 + a[1][0]
17. for i=1 to n-1
18. dp[0][i%2] = MIN(dp[0][(i-1)%2],dp[1][(i-1)%2]+t[1][i-1])+a[0][i]
19. dp[1][i%2] = MIN(dp[1][(i-1)%2],dp[0][(i-1)%2]+t[0][i-1])+a[1][i]
20. ans = MIN(dp[0][(n-1)%2]+x1,dp[1][(n-1)%2]+x2))
21. write ans

SOURCECODE

```
#include <cstdio>
#include <algorithm>
using namespace std;
int dp[2][2];
int n,x1,x2;
int a[2][1024], t[2][1024];
template<typename T>
T getNum(){
    T res=0;
    char c;
    while(1){
        c = getchar_unlocked();
        if(c==' '||c=='\n') continue;
        else break;
    }
    res=c-'0';
    while(1){
        c = getchar_unlocked();
        if(c>='0' && c<='9') res=10*res+c-'0';
        else break;
    }
    return res;
}
int main(){
    while(scanf("%d",&n)!=EOF){
        dp[0][0] = getNum<int>();
        dp[1][0] = getNum<int>();
        for(int j=0;j<2;j++)
            for(int i=0; i<n ;i++) a[j][i]=getNum<int>();
        for(int j=0;j<2;j++)
            for(int i=0;i<n-1;i++) t[j][i]=getNum<int>();
        x1 = getNum<int>();
        x2 = getNum<int>();
        dp[0][0] += a[0][0];
        dp[1][0] += a[1][0];
        for(int i=1;i<n;i++){
            dp[0][i%2] = min(dp[0][(i-1)%2],
                  dp[1][(i-1)%2]+t[1][i-1])+a[0][i];
            dp[1][i%2] = min(dp[1][(i-1)%2],
                  dp[0][(i-1)%2]+t[0][i-1])+a[1][i];
        }
        printf("%d\n",
                  min(dp[0][(n-1)%2]+x1,dp[1][(n-1)%2]+x2));
    }
    return res;
}
```

SUBMISSIONS

SUBMISSION # 22748039

PROBLEM:	2769 - Assembly Line
ANSWER:	**Accepted**
LANGUAGE:	C++17 (g++ 7.3.0, -std=c++17 -O2 -lm) [+0s]
RUNTIME:	0.000s
FILE SIZE:	990 Bytes
MEMORY:	-
CODE GOLF:	0 characters (-1093 than the median)
SUBMISSION:	5/10/21, 12:06:22 AM