SE SW Competency Task Force(SSCTF)

Class - 4

SSC Task Force

Anju Bala

Arabinda Verma

Chetan Sai Kumar T

Prathap HS

Vikas Agrawal

Veeraj S Khokale

Sasikumar Sathasivam

- # Toll Gate Problem
  - **(Similar to Mar’16 Adv Test)**
  - **Brainstorming session and approach to resolve**

Class-4

Practice: Advance Test –TollGate– Mar'16

Please find the **minimum cost** to travel from Source to Destination location with multiple toll gates across

There are challenges at each toll gate to minimize the cost.

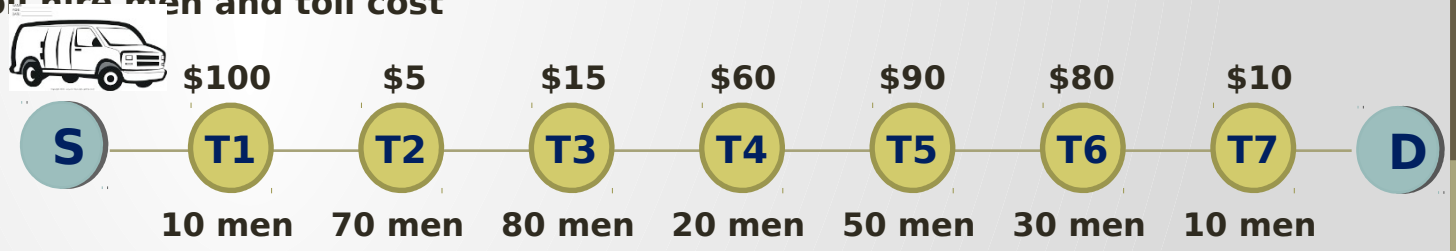
- One can either choose to **pay the toll** or
- One can battle at the toll gate to **avoid paying** by having his own set of men's they travel with them (initially zero) or
- One can **pay double the toll cost and hire all the men** at the each tolls for the next toll to battle and avoid toll cost,
- If you choose to battle at particular toll only if you can have no.of.. hired men is more than the count hired men at respective toll gate.

Note: Each hired men can battle for 3 times only

- For each battle you will lose equal no.of.men with you as well as available in the toll gate . Rest of them will lose 1 round of battle irrespective of they are alive or not. After 3 battle they will not survive. *If you have 10 men with you and toll no. of. Toll men is 8, then you lose 8 men in battle and remaining 2 men lost 1 round of battle and hence they can be available for 2 more rounds only.*

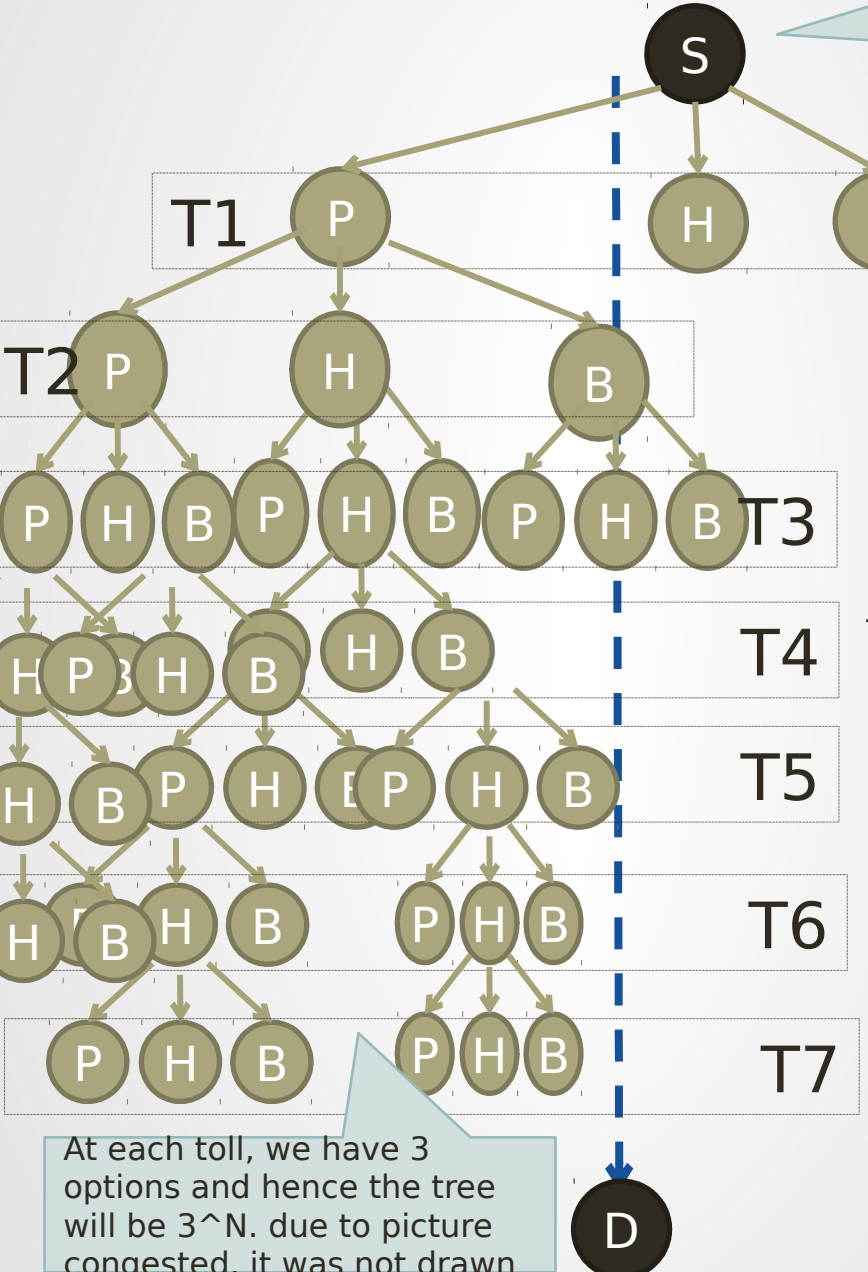
Ex:

7 //toll no.of.tolls
10 100 //toll hire men and toll cost
70 5
80 15
20 60
50 90
30 80
10 10
Min cost: 150



Practice

TollGate – Approach



Approach: At each toll, we have 3 options. Hence we need to explore all 3 paths. **This can be done with traversal of tree in preorder or dfs.** Both Pay and Hire, can move forward as we traverse, but we can proceed battle traversal only if it succeeds.

- Time complexity for 1 test case is 3^N
- if N is 20, then time complexity is 34,867,84410

How to reduce the computing cycle?

- All battle will not happen as we may not have enough men with us to fight at all tolls.
- Also there is no point in traversing all the path if already computed cost is minimum than new computed cost and hence it can be abandon mid way

T1 - Tollgate1

P Pay toll

H Hire and pay double toll

B Battle , do not pay toll

TollGate – Problem solving approach

```
int N, t_cost [22], t_hire[22], min_cost = 1000000;
```

```
void dfs (int tp, int cc)
{
if (tp == N-1) //base condition to check last toll g
   {
     cc += tc[tp];
     if ( cc < min_cost) min_cost = cc;
     return;
   }

    dfs(tp+1, cc+tc[tp]); //toll pay option

}
```

Let us assume this is code that will help to traverse **toll pay** option only.

```
void dfs (int tp, int bp3, int bp2, int bp1, int cc)
{

   if (tp == N-1) //base condition to check last toll
   {
     cc += tc[tp];
     if ( cc < min_cost) min_cost = cc;
     return;
   }

   dfs(tp+1, bp3+th[tp], bp2, bp1 , cc+2*tc[tp]); //toll hire
option

}
```

Similarly this will be the code that will help to traverse **toll hire and double toll pay option** only.

oll position**, cc-** current cost**, tc-**toll cost, **th**-toll hire, **bp1-bp2-bp3** – battle poo

TollGate – Problem solving approach

```
int N,  t_cost  [22], t_hire[22], min_cost = 1000000;
void dfs(int tp, int bp3, int bp2, int bp1, int cc)
{
    int tot_bp = bp3 + bp2 + bp1;

    if (tp == N-1) //base condition to check last toll ga
    {
         if ( tot_bp < th[tp]) cc += tc[tp];
         if ( cc < min_cost) min_cost = cc;
     return;
    }


    if ( tot_bp >= th[tp] ) //toll battle option
    {
     if ( th[tp] > bp2 + bp1 )
     {
        bp3 = tot_bp - th[tp];
        bp1 = bp2 = 0;
     }
     else if ( th[tp] > bp1 )
     {
        bp2 = (bp1+bp2) - th[tp];
        bp1 = 0;
     }
     dfs(tp+1, 0, bp3 , bp2, cc); //note: pool3 is zero, pool3 becomes
pool2 and pool2 as pool1
    }
}
```

This piece of code for **toll battle** option only.

toll position**, cc-** current cost**, tc-**toll cost, **th**-toll hire, **bp1-bp2-bp3** – battle po

TollGate – Problem solving approach

```
int N,  t_cost  [22], t_hire[22], min_cost = 1000000;
  void dfs(int tp, int bp3, int bp2, int bp1, int cc)
  {
      int tot_bp = bp3 + bp2 + bp1;

      if (tp == N-1) //base condition to check last toll ga
      {
            if ( tot_bp < th[tp]) cc += tc[tp];
       if ( cc < min_cost) min_cost = cc;
       return;
      }

       dfs(tp+1, bp3 , bp2, bp1 , cc+tc[tp]); //toll pay option
       dfs(tp+1, bp3+th[tp], bp2, bp1 , cc+2*tc[tp]); //toll hire
  option

      if ( tot_bp >= th[tp] ) //toll battle option
      {
       if ( th[tp] > bp2 + bp1 )
       {
       bp3 = tot_bp - th[tp];
       bp1 = bp2 = 0;
       }
       else if ( th[tp] > bp1 )
       {
       bp2 = (bp1+bp2) - th[tp];
       bp1 = 0;
       }
       dfs(tp+1, 0, bp3 , bp2, cc); // note: pool3 is zero, pool3 becomes
  pool2 and pool2 as pool1
      }
  }
```

Merge all 3 codes.
Is this efficient?
Time complexity for N=20 is 3^N

- toll position, **cc-** current cost, **tc-**toll cost, **th**-toll hire, **bp1-bp2-bp3** – battle p

TollGate – Problem solving approach

```
int N,  t_cost  [22], t_hire[22], min_cost = 1000000;
  void dfs(int tp, int bp3, int bp2, int bp1, int cc)
  {
      int tot_bp = bp3 + bp2 + bp1;
      if (cc > min_cost) return; // condition important to avoid unnecessary cpu
  cycle
      if (tp == N-1) //base condition to check last toll
      {
            if ( tot_bp < th[tp] ) cc += tc[tp];
       if ( cc < min_cost) min_cost = cc;
       return;
      }

       dfs(tp+1, bp3 , bp2, bp1 , cc+tc[tp]); //toll pay option
       dfs(tp+1, bp3+th[tp], bp2, bp1 , cc+2*tc[tp]); //toll hire
  option

      if ( tot_bp >= th[tp] ) //toll battle option
      {
       if (th[tp] > bp2 + bp1 )
       {
       bp3 = tot_bp - th[tp];
       bp1 = bp2 = 0;
       }
       else if (th[tp] > bp1 )
       {
       bp2 = (bp1+bp2) - th[tp];
       bp1 = 0;
       }
       dfs(tp+1, 0, bp3 , bp2, cc); // note: pool3 is zero, pool3 becomes
  pool1 and pool2 as pool
      }
```

This condition will avoid **unnecessary traversal** if the cost is going more than already computed min cost.

Rotating Battle pool members 3 to 2 and 2 to 1 pool

tp-toll position, **cc**-current cost, **tc**-toll cost, **th**-toll hire, **bp1-bp2-bp3** – battle p

TollGate – Main function

```
#include <stdio.h>
#include<time.h>
// no.of.toll gate(between 5 and 20, cost at toll gate, total hire available at
tollgate, minimum cost
int N, tc [22], th[22], min_cost = 1000000;
void dfs(int tp, int bp3, int bp2, int bp1, int cc);
int main()
{
      int i, TC;
      clock_t start, end;
      double cpu_time_used;

      printf("No.of.TC? "); scanf("%d", &TC);
      start = clock();
      while( TC-- )
      {
           scanf("%d", &N);
           for ( i = 0; i < N; ++i)
            scanf("%d %d", &th[i], & tc [i]);


           dfs(0, 0, 0, 0, 0);
           printf("\nMinCost= %d\n\n", min_cost );
           min_cost = 1000000; //some large number
      }
      end = clock();
      cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
      printf("fun() took %f seconds to execute \n", cpu_time_used);
      getch();
     return 0;
}
```

Mar_Adv_TollGate-sasi-v3.c.txt

TollGate_input.txt

TollGate-Sasi(AdvSWMar16).7z

- toll position, **cc-** current cost, **tc-**toll cost, **th**-toll hire, **bp1-bp2-bp3** – battle p

TollGate – Output

```
No.of.TC? 5
7
10 100
70 5
80 15
20 60
50 90
30 80
10 10

MinCost= 150


9
600 800
300 400
300 400
1000 400
300 600
100 300
600 300
600 500
1000 300

MinCost= 3000


11
1000 10
700 900
400 500
300 10
900 900
300 10
50 900
50 900
700 900
500 900
50 10

MinCost= 2370
```

```
20
896 546
543 216
454 310
408 367
40 602
252 582
954 627
850 234
763 479
232 278
301 538
528 508
936 154
629 443
758 336
432 700
882 256
278 738
517 882
317 136

MinCost= 4721


20
410 610
831 909
675 629
421 774
386 869
544 219
492 414
996 557
499 482
231 285
804 978
304 881
489 911
75 315
927 648
252 914
330 396
937 133
495 882
813 717

MinCost= 8231

fun() took 0.062000 seconds to execute
```

TollGate – cpp

```cpp
#include <iostream>
int N, cc[25], t[25], min_cost = 10000007;

void dfs(int p, int a, int b, int c, int cost)
{
     int asum = a+b+c;
     if (cost > min_cost) return;
     if (p == N-1)
     {
          if (asum < t[p]) cost += cc[p];
          if (cost < min_cost) min_cost = cost;
          return;
     }
     dfs(p+1, a, b, c, cost+cc[p]);
     dfs(p+1, a+t[p], b, c, cost+2*cc[p]);
     if (asum >= t[p])
     {
          if (t[p] > b+c) a = asum-t[p];
          if (t[p] > c) b = t[p]-c>=b ? 0 : b-t[p]+c;
          dfs(p+1, 0, a, b, cost);
     }
}
int main()
{
     std::cin >> N;
     for (int i = 0; i < N; ++i)
          std::cin >> t[i] >> cc[i];
     dfs(0, 0, 0, 0, 0);
     std::cout << min_cost << std::endl;
     return 0;
}
```

Thank You