



Competitions

- Overview
- Copilot Opportunities
- Design
- Development
- UI Development
- QA and Maintenance
- CloudSpokes
- Algorithm
- High School
- The Digital Run
- Submit & Review

TopCoder Networks

- Events
- Statistics
- Tutorials
- Forums
- Surveys
- My TopCoder
- Help Center
- About TopCoder



Member Search:

Handle: Go

Advanced Search

Dashboard > TopCoder Competitions > ... > Algorithm Problem Set
Analysis > SRM 626

TopCoder Competitions

SRM 626

View

Attachments (10)

Info

Browse Space

Added by [[rng_58]], last edited by vexorian on Jul 05, 2014 ([view change](#))

Labels: (None) EDIT

Single Round Match 626

Saturday, June 28th, 2014

Archive

[Match Overview](#)[Discuss this match](#)

Match summary

The Problems

[SumOfPower](#) | [FixedDiceGameDiv2](#) | [NegativeGraphDiv2](#) | [FixedDiceGameDiv1](#) | [NegativeGraphDiv1](#) | [ReflectiveRectangle](#)

NegativeGraphDiv1

Rate It

Discuss it

Used as: Division One - Level Two:

Value	600
Submission Rate	110 / 739 (14.88%)
Success Rate	60 / 110 (54.55%)
High Score	tourist for 562.70 points (7 mins 24 secs)
Average Score	351.68 (for 60 correct submissions)

Because of the high constraints for the number of charges, we will need a very different approach than the [division 2 version](#). There are ideas we can take from it, however. We will imagine 0-weight "negative" edges between u and itself so that we can use "exactly" all the charges instead of "at most". It is also useful to know how to solve the special graph (k, u) .

Shortest paths with just one negative edge

The idea we will use is to reduce the problem into something different and easier to optimize.

Consider only the negative edges we use. Imagine we used negative edges $(x_0 \rightarrow x_1), (x_2 \rightarrow x_3), \dots, (x_{k-1} \rightarrow x_k)$. Then we can imagine that the total path involves moving from vertex 1 to x_0 (using a path of non-negative edges), then use the negative edge $(x_0 \rightarrow x_1)$ then move from x_1 to x_2 , use the negative edge $(x_2 \rightarrow x_3)$ and so and so until we do a normal path between x_{k-2} to x_{k-1} , using the negative edge $(x_{k-1} \rightarrow x_k)$ and finally we use a normal path from x_k to N .

Imagine that sequence a one of some paths: A path from 1 to x_1 , a path from x_1 to x_3, \dots , a path from x_{k-2} to N . In each of these paths we use exactly one negative edge.

How about this reduction: First calculate all the minimum distances from each u to each v using exactly one negative edge. We will call this $A[u][v]$. Imagine a new complete graph that had edges of weight $A[u][v]$. (Note that $A[u][u] \leq 0$ for every u , because we are assuming a negative loop of length 0 exists for each vertex). The problem becomes to find the minimum distance between 1 and N using exactly K edges. For convenience we'll use $K = \text{charges}$. Note that we can return to a vertex multiple times if necessary.

Exactly K edges

The minimum path using exactly K edges when K can be very large is a known problem that can be solved using a matrix exponentiation-esque approach. Consider $D_1 = A$, the matrix that calculates the minimum distances from each u to each v using exactly one edge. We can use this matrix to calculate D_2 . We will use notation $D_2 = D_1 \star D_1 = A \star A$

Imagine we want to calculate $A \star B$. The idea is that we will find an intermediary vertex k , we can try all $O(n)$ vertices k : $(A \star B)[i][j] = \min(A[i][k] + B[k][j], 1 \leq k \leq n)$. $A \star B$ can be calculated in $O(n^3)$ time. We can use this logic to calculate $A \star A$.

Note that we can use $D_K = D_{K-1} \star A = A \star A \star \dots \star A$ (K times). We will represent this as A^K . It turns out that the \star operation is associative. So $A^t = A^{\frac{t}{2}} \star A^{\frac{t}{2}}$ when t is even. This allows us to calculate A^K using only $O(\log(K))$ \star operations. In total: $O(n^3 \log(K))$.

For example, with $K = 13 = 8 + 4 + 1$ we have $A^{13} = A^8 \star A^4 \star A^1 = A^4 \star A^4 \star A^1$, $A^4 = A^2 \star A^2$ and $A^2 = A \star A$. We can use the [exponentiation by squaring](#) algorithm.

Code

```

    }
}
}
if (charges == 0) {
    return dist[0][N-1];
}
matrix A(N, vector<long>(N));
for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        A[i][j] = dist[i + N][j];
    }
    // adding these fake loops will make it so the power yields the result
    // of using at most k edges instead of exactly k edges:
    A[i][i] = min(A[i][i], 0L);
}
int k = charges;

// D = A ^ k
// Exponentiation by squaring, using The * operation
// As the base for the exponentiation by squaring, we need an Identity
// matrix. For this special * operation, imagine the matrix for moving
// from each u to each v using ZERO edges. When u != v this is impossible
// so we can use INF, when u = v, it is possible (0 cost).
matrix D(N, vector<long>(N, INF));
for (int i = 0; i < N; i++) {
    D[i][i] = 0;
}

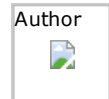
while (k > 0) {
    if (k % 2 == 1) {
        D = combineMatrices(D, A);
    }
    A = combineMatrices(A, A);
}

```

Alternative solutions and additional comments.

<Place your comments here>

Next problem: [ReflectiveRectangle](#)



By **vexorian**

TopCoder Member

Editorial feedback	Choose
I liked it.	<input checked="" type="radio"/>
I didn't like it.	<input checked="" type="radio"/>

Comments ([Hide Comments](#))

As it's in the Wiki, there's a possibility to improve it. It can be language correction, wording improvement or additional explanation in some parts, your additional comments, description of alternative solutions, etc. If you want to improve the wording of editorial writer or correct some language error, please feel free to put your change over the original text. And if you wish to add a comment or describe another approach, there's a section for this at the bottom of each problem.

Before editing, please be sure to check the [guidelines](#).

You can also add a comment in this comment section. When posting a comment thread, make sure to specify the problem you are talking about.

The editorial uses [mathjax](#) to render formulas and math symbols, if you have issues with the formatting of math contents or they don't show up, try right clicking the space that contains them and change the render mode in the popup menu.


 Posted by vexorian at [Jul 02, 2014 11:48](#) | [Reply To This](#)

As it's in the Wiki, there's a possibility to improve it. It can be language correction, wording improvement or additional explanation in some parts, your additional comments, description of alternative solutions, etc. If you want to improve the wording of editorial writer or correct some language error, please feel free to put your change over the original text. And if you wish to add a comment or describe another approach, there's a section for this at the bottom of each problem.


Before editing, please be sure to check the [guidelines](#).

You can also add a comment in this comment section. When posting a comment thread, make sure to specify the problem you are talking about.

The editorial uses [mathjax](#) to render formulas and math symbols, if you have issues with the formatting of math contents or they don't show up, try right clicking the space that contains them and change the render mode in the popup menu.


 Posted by vexorian at [Jul 02, 2014 11:48](#) | [Reply To This](#)

n


 Posted by flyman3046 at Jul 02, 2014 20:49 Updated by flyman3046 | [Reply To This](#)

Complexity of Div 2 1000 problem theoratically is still pretty high. $charges * N * M > 1e8$. How can we be sure that it wont time out ?

I mostly see that if the number of operations $\leq 1e8$ then it would not time out.

 Posted by nilmish.iit at [Jul 02, 2014 23:08](#) | [Reply To This](#)

In topcoder $\leq 1e9$ tends to be fine actually, specially in case of this problem with simple operations. C++ takes 0.5 seconds in worst case and Java is usually 2x slower so it should be okay too.

 Posted by vexorian at [Jul 03, 2014 00:43](#) | [Reply To This](#)

About FixedDiceGameDiv1, another idea to calculate number of every possible score and then calculate probability finally. The codes like this:

```
int flag_a = 0;
for (int i=1; i<=b; i++) an[flag_a][i] = 1;
for (int i=1; i<a; i++) {
    for (int j=1; j<=b; j++)
```

Unknown macro: { for (int k=0; k<2501; k++) an[1-flag_a][j+k] += an[flag_a][k]; }

```
flag_a = 1 - flag_a;
for (int j=0; j<2501; j++) an[ 1-flag_a ][j] = 0;
}
```

```
double total_case_a = 0;
for (int i=0; i<2501; i++)
```

Unknown macro: { total_case_a += an[flag_a][i]; }

```
for (int i=0; i<2501; i++)
```

Unknown macro: { an[flag_a][i] = an[flag_a][i] / total_case_a; }

The result can be quite huge and seems only double can store it. The answer is not same with the standard solution when the parameters are big. It should due to accuracy lost of double type. My question is how can we be sure there is no accuracy lost in the standard solution?

 Posted by houxiang at [Jul 03, 2014 02:48](#) | [Reply To This](#)

In the problem ReflectiveRectangle I didn't get the line "We can find the point with $x_1=1$ that has b bounces: $y_1=1+b$ ". What are x_1 and y_1 here?

I get a feeling I'm missing something trivial, if it is so, please forgive me.

 Posted by adityavikram at [Jul 03, 2014 06:44](#) | [Reply To This](#)

They are just the names of the point that has b bounces and its x coordinate is 1. I should have probably added numbers to the grids so that it is easy to see that the grid makes a coordinate system though.

 Posted by vexorian at [Jul 03, 2014 10:42](#) | [Reply To This](#)


In the editorial of NegativeGraphDiv2 I haven't got this portion...

```
for (int k = 1; k <= charges; k++) { // O(charges)
for (int i = 0; i < N; i++) { // O(charges * N )
xdist[k][i] = xdist[k-1][i]; //"at most" k charges
// (*This is equivalent to a 0-weight "negative" loop i->i)
```

How is distance from node 'i' to node 'N-1' using at most K charges equivalent to distance from node 'i' to node 'N-1' using at most $K-1$ charges ??? Would anybody give me the answer please ???


 Posted by mukitmbks25 at [Jul 05, 2014 10:52](#) | [Reply To This](#)

It isn't, but the distance using at most K charges shall be at most equal to the distance with $K-1$ charges. Because if there is a path using $K-1$ charges and it is very small, you can still use that same path when the condition is "at most K ".


 Posted by vexorian at [Jul 05, 2014 11:58](#) | [Reply To This](#)

In NegativeGraphDiv1:

(Note that $A[u][u]=0$, for every u). Is it corrent? Isn't $A[u][u]$ negative when there is self-loop in u ?

 Posted by sluki at [Jul 05, 2014 13:58](#) | [Reply To This](#)


Changed that part.

 Posted by vexorian at [Jul 05, 2014 16:44](#) | [Reply To This](#)

In NegativeGraphDiv2 why we can not use Dijkstra after making new graph ?

I think negative edges can not be problem in our Dijkstra, because whenever we use a negative edge, we can not update any previous node. I mean, the reason that make Dijkstra wrong on a graph with negative edge, isn't true here.

 Posted by arabpour.m at [Jul 25, 2014 14:43](#) | [Reply To This](#)

Can someone please explain the O  solution for SumOfPower div 2 250 pt. problem?

```
int ans = 0;
```

```
for(int i = 0; i < N; ++i)
```

```
Unknown macro: {  ans += (i+1)*(N - i)*array[i]; }
```

Didn't understand that at all.

Thanks.

 Posted by blizzard_one at [Aug 26, 2014 02:59](#) | [Reply To This](#)

 [Add Comment](#)