T212 - Food Kangaroo

Kelvin Chow {Lrt1088}

2021-04-18

香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Background

Author: Kelvin Chow

Setter: Anson Ho, Charlie Li, Gabriel Liu, Kelvin Chow, Percy Wong and Tony Wong

Statistics

Task	Attempts	Max	Mean	Std Dev
T212 - Food Kangaroo	51	100	16.098	26.847

Subtasks					
6: 30	11: 22	15: 7	24: 4	22: 5	22: 4

First solved by **s16210** at **1:16**

SUBTASKS

For all cases:
$2 \le N \le 2 \times 10^5$
$0 \le H_F, D \le 2 \times 10^5$
$0 \le H_T \le 500$
$1 \le t_j \le 10^9$ (for all $1 \le j \le H_T$)
$1 \le R_k \le 10^9$ (for all $1 \le k \le D$)

	Points	Constraints
1	6	$H_T = 0$
2	11	$D = 1$
3	15	$2 \le N \le 50$ $H_F = 0$ $0 \le D \le 10$ $0 \le H_T \le 50$
4	24	$H_F = 0$
5	22	There is no way to earn infinitely many Bytecoins.
6	22	No additional constraints.

SCORING

In subtasks **except Subtask 5**, you can obtain a partial score. For each subtask (except Subtask 5):

- You score 100% if your output is correct in all test cases within the subtask; otherwise
- You score 30% if the first line of your output is correct in all test cases within the subtask; otherwise
- You score 0.

Problem

Given a Simple Graph (**Not guaranteed Connected)**, with H_F edge with 0 cost, and H_T edges with cost.

There D orders which delivered from X_k to Y_k, which have rewards of R_k, all orders can receive infinitely.

Find if there are a way to earn infinite amount of money, or the maximum of money can Bob earn.

Subtask 1

6 points: All edges have 0 cost.

The cost travel to any city is 0.

If there are a order, Bob can receive it infinitely without any cost.

If D > 0, just output “YES”, right?

The graph **not guaranteed Connected**, which mean orders may can’t be reach.

If order order can’t be reach, Bob can’t earn any money.

Subtask 1

Solution:

- Perform a DFS/BFS to check the connectivity to city 1.
- If any X_k is connected to city 1, output “YES”, else output “NO\n0”.

Time Complexity: $O(N+H_F)$ for DFS + $O(D)$ for checking X_k.

Subtask 2

11 points: Only 1 order.

Only 1 order we need to be considered.

In what condition, Bob can earn infinite amount of money?

Everytime after Bob delivery the order, he need to go back to X_k from Y_k.

The cost of a delivery is the shortest path from X_k to Y_k.

The cost of going back to X_k from Y_k is the shortest path from X_k to Y_k also.

If the Reward can cover both trip, that mean Bob can use this order infinitely.

Remind that X_k can be reached at first to let all this happen.

Subtask 2

If Bob can't earn infinite amount of money, what is the maximum?

Bob need to go to Xk from city 1 to collect the order, and then go to Y_k to deliver it, finally he get the reward R_k.

Let S[i][j] = the shortest path i to j.

The maximum he can earn = R_k - S[1][X_k] - S[X_k][Y_k]

Remind that if Rk < (S[1][X_k] + S[X_k][Y_k]), Bob can choose to not do anything,

i.e earn 0.

Subtask 2

Solution:

- Find the shortest path from 1 to X_k.
- If Xk isn't connect to 1,
  - output "NO\n0".
- Else
  - find the shortest path from X_k to Y_k.
  - If Rk > $S[X_k][Y_k]$*2
    - output "YES"
  - Else
    - output "NO\n", max(0, R_k - $S[1][X_k]$ - $S[X_k][Y_k]$)

Subtask 2

Time Complexity: depend of what shortest path algorithm you used.

If Dijkstra was used, $O((H_F+H_T)+NlogN) * 2 = O((H_F+H_T)+NlogN)$

Subtask 3

15 points: So many constrain.

It’s seems like some kind of exhaustion.

Actually, there are a lot of different approaches.

I’ll talk about the one that I can think of.

3	15	$2 \leq N \leq 50$ $H_F = 0$ $0 \leq D \leq 10$ $0 \leq H_T \leq 50$

Subtask 3

(From now, I'll consider cities that connect to city 1 only)

Let P_1, P_2, ..., P_i = a list of orders P that Bob must deliver one by one.

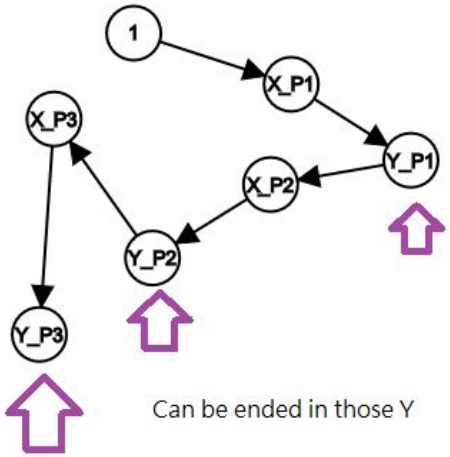
How can Bob deliver all of them optimally?

Bob should go from 1 to X_{P1} via the shortest path, and then X_{P1} to Y_{P1}, and then Y_{P1} to X_{P2} ... until arriving Y_{Pi}.

If we guarantee Bob can not earn money infinitely, we can just try all permutation of the orders.

Be aware that not every order needed to be delivered, so every intermediate steps also need to be considered.

Subtask 3



Subtask 3

How to know if Bob can earn infinite money?

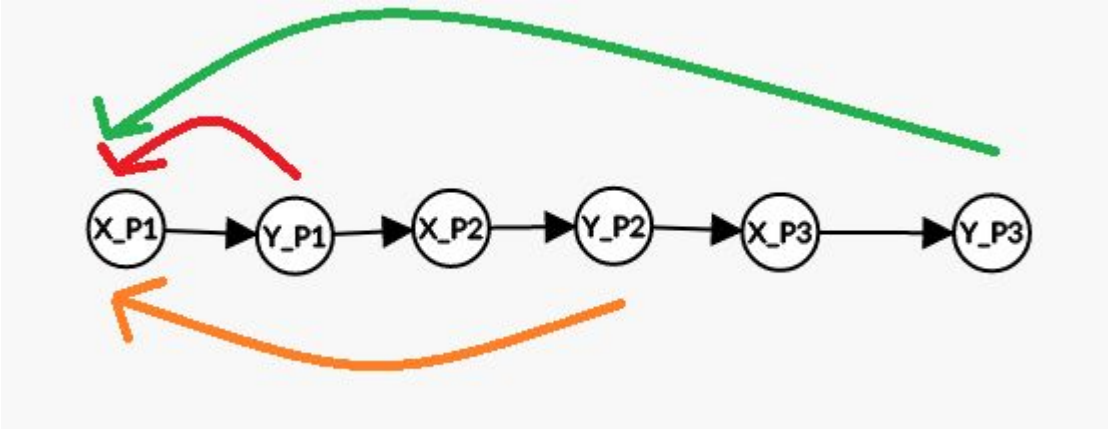
Let P_1, P_2, ..., P_i = a list of orders P that Bob must deliver one by one again, but after P_i will go back to P_1.

Let exclude the trip from city 1 to X_{P1}, i.e just consider X_{P1} to Y_{P1}, Y_{P1} to X_{P2}, X_{P2} to Y_{P2}, ..., Y_{Pi} to X_{P1}.

If the total reward is greater than the total cost, that mean Bob can earn money infinitely in this order.

Again consider permutations, and every intermediate steps.

Subtask 3



Subtask 3

OK, but how?

We need to find the shortest path from city 1, X_k and Y_k to every city first.

We can use Dijkstra algorithm multiple times since D <= 10.

Or just use Floyd-Warshall algorithm to find the shortest path for every pair of cities.

And then for every permutations, carefully calculate the cost to find can Bob earn money infinitely or the maximum Bob can earn.

Time complexity: $(O(N^3)$ or $O((H_T+NlogN)*D))+O(D!)$

Subtask 4

15+24 points: $H_F = 0$

That mean total number of edge is $H_T <= 500$, and a edge can connect two cities only.

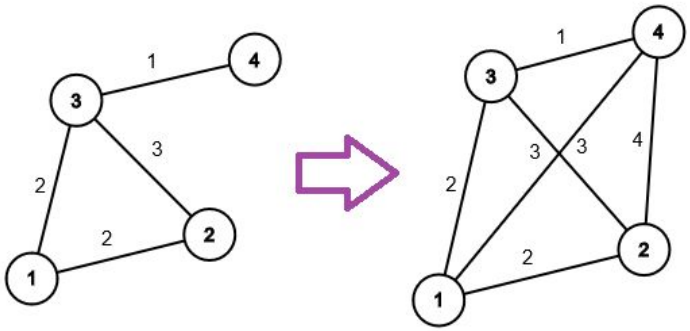
Which mean we only need to consider at most ~1000 cities only since other cities didn't connect to any cities because of out of edges.

We can apply Floyd-Warshall algorithm to find the shortest path for every pair of cities.

OK, what's next?

Subtask 4

After completing the Floyd-Warshall algorithm, we actually converted the original graph to a complete graph, and the cost of each edges of this complete graph is the shortest path of each pair.



Subtask 4

What if we add some order from city X to Y with reward R?

The cost of travel from X to Y is reduced by R because Bob can deliver the order when travelling from X to Y.

We now constructed a “directional complete graph”(is it the right way to call it?).

Be careful Bob can only take a order once and there may have multiple orders deliver from X to Y.

Subtask 4

Let's change the direction of what's the meaning of "the maximum Bytecoins Bob can earn".

It's equivalent to opposite of "the minimum Bytecoins Bob can lose".

What's the meaning of finding the shortest path of the "directional complete graph" we constructed from city 1 to city i?

Isn't it just mean the minimum Bytecoins Bob can lose travelling from city 1 to city i!!!

Subtask 4

(from Graph (II))

	Type	Negative edge?	Time Complexity	
Dijkstra's Algorithm	single source	not support	O(ElogE + V)	
Bellman-Ford Algorithm	single source	support	O(VE)	
SPFA	single source	support	Average Case: O(E) [random graph]	Worst Case: O(VE)
Floyd-Warshall Algorithm	all-pairs	support	$O(V^3)$	

Subtask 4

Since R_k may greater $S[X_k][Y_k]$ which mean there may be negative edges.

Only Bellman-Ford Algorithm and Floyd-Warshall Algorithm can be apply.
(SPFA is a Improvement of Bellman-Ford Algorithm)

Let's talk about them one by one.

Subtask 4

For Bellman-Ford Algorithm, we can plainly apply it to the “directional complete graph”.

After the completion of it, the minimum value of the shortest path from city 1 to i is the answer. i.e. ans = -min(d[i]) where d[i] = the shortest path from city 1 to i.

What if a negative cycles detected?

That mean Bob can lose money negative infinitely, i.e Bob can earn money positive infinitely.

Subtask 4

The time complexity of Bellman-Ford Algorithm is O(VE)

By the observation of previous slide, the number vertex = $O(H_T)$.

The number of edge of a complete graph = $O(V^2)$

The time complexity for this task = $O(H_T^3)$

Subtask 4

For Floyd-Warshall Algorithm, we can plainly apply it to the “directional complete graph” too.

After the completion of it, the minimum value of the shortest path from city 1 to i is the answer. i.e. ans = -min(dp[1][j]) where dp[1][j] = the shortest path from city 1 to j.

Floyd-Warshall can also be used to detect negative cycles.

If any diagonal i.e. dp[i][i] < 0, that mean somehow Bob can gain money from city i to i.

That’s mean negative cycle detected.

Subtask 4

The time complexity of Floyd-Warshall Algorithm is $O(V^3)$

By the observation of previous slide, the number vertex = $O(H_T)$.

The time complexity for this task = $O(H_T^3)$

Same as Bellman-Ford Algorithm

Subtask 4

The time complexity = $O(N+H_T)$ for checking connectivity + $O(H_T^3)$ for the first Floyd-Warshall Algorithm + $O(D+H_T^3)$ for the final part.

Please be careful of the connectivity.

Subtask 6

Let’s jump to full solution.

Now we have a lot of edges now, that mean we cannot consider ~1000 cities only, or can we?

Subtask 6

Let's consider the sample test case.

Now the order is deliver from 2 to 5.

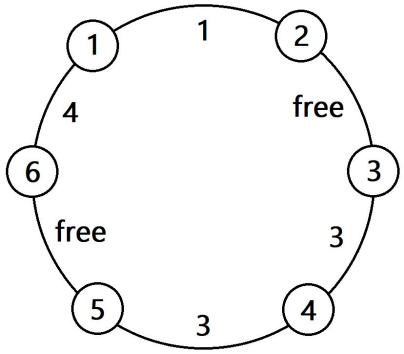
What if we change it from 3 to 6?

The answer won't change, why?

Note that both 2&3 and 5&6 are connected by free roads.

Actually as long as those cities are connected by some free roads, the starting point and ending point doesn't matter.

Input	Output
6 2 4 1 2 3 6 5 1 2 1 3 4 3 4 5 3 6 1 4 2 5 7	NO 1



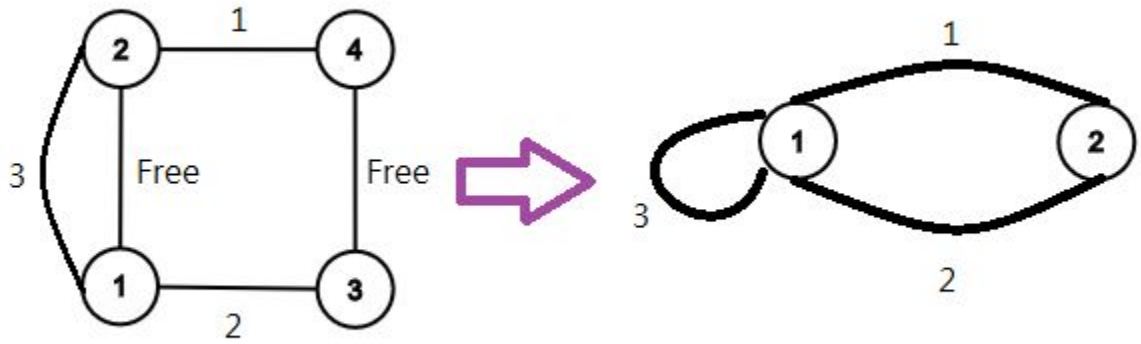
Subtask 6

By this observation, we can now shrink the cities connected by some free roads to a single node, and giving a id to each node.

And then we can reconstruct all tolled roads and orders to the shrunken version.

Notice that we now may introducing repeated edges and/or self loop into shrunken graph.

Subtask 6



Subtask 6

Since H_T <= 500 still, we can use the same solution of subtask 4 on the shrunken graph.

The time complexity = $O(N+(H_F+H_T))$ for checking connectivity + $O(N+(H_F+H_T)+H_T+D)$ for shrinking the graph + $O(H_T^3)$ for the first Floyd-Warshall Algorithm + $O(D+H_T^3)$ for the final part.

Please please please be careful of the connectivity.

Subtask 6

“Why I do exactly what you’d said but TLE???”

Actually H_T <= 500 is quite tight for a ~$O(H_T^3)$ solution.

If you wrote bad code, you may need some constant time optimization.

For example, I’d said there are roughly ~1000 nodes, actually we don’t need to consider that much.

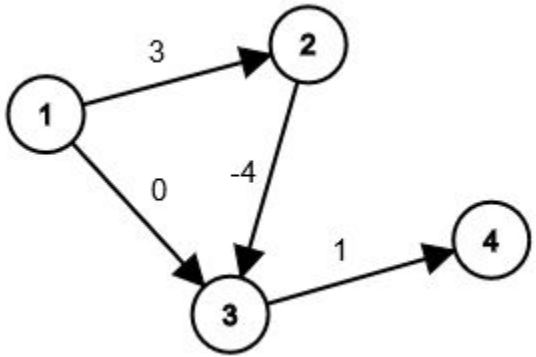
In worst case, those toll roads from a tree, that mean there are at most 501 nodes. There may have a constant time optimization of $2^3 = 8$.

Subtask 6

“What if I use Dijkstra’s Algorithm for the final part?”

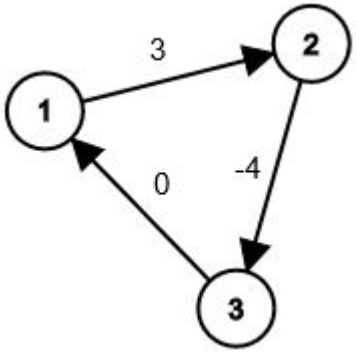
Depend of how you implement the algorithm.

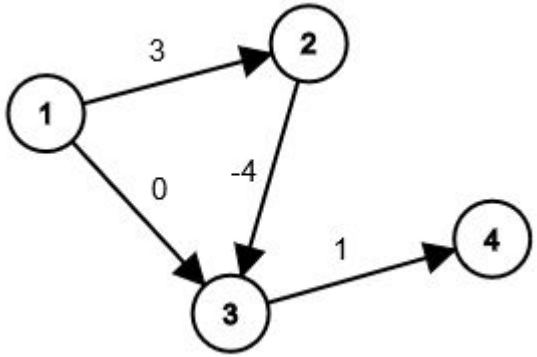
If a visited node won’t be visited again, then it will WA.



Subtask 6

If a visited node can be visited again, then it may TLE.





Subtask 5 and 30% score

The purpose of setting them is because we want to reward someone can come up with full solution but don't know it can determined YES or NO, or find the maximum value. For Subtask 5, your solution pass this too if it fail when there are negative cycle.