Homework 7, CSE 232

Due October 22 Note: On most of the problem sets through the semester, I'll put a horizontal line with "Optional" under it. Any problems below this section are encouraged - I think they're interesting and will help you learn the subject - but not necessary to complete in order to get credit for the homework.

Problem 1

This week's problem is from the Timus Online Judge #1210 - Kind Spirits. Here's the problem statement, slightly rewritten for clarity:

> Ivanushka the Fool lives at the planet of 0th-level. It's very unpleasant to live there. An awful climate, 80 hours working week, bad food. He, as well as every inhabitant of his planet, dreams to get to a planet of N-th level. To the paradise.
>
> At each of the i-th level planets there are several hyperspace transfers to some of the (i+1)-st level planets (but there are no reverse paths). Every transfer is guarded by a spirit. The spirits are usually evil: they demand many galactic bank-notes for each transfer. You know, everyone wants to go to a higher level planet, and one has to pay for the pleasure. More than Ivanushka can even imagine. However, extraordinary situations like a lack of a labor-force at one of the higher level planets sometimes happen, and then the spirits - the guards of the transfers — become kind. Sometimes they give galactic bank-notes themselves if only someone goes to their planets.
>
> In order to embody his dream of visiting a heavenly planet Ivanushka has done two things. First of all, he has borrowed a complete map of the Universe. It's written on the map how much the spirits demand or give for a transfer from this or that planet to another one of the next higher level. Secondly, he has hired a staff of young talented programmers in order that they will help him to draw the way on the map from his planet to the one of Nth level to find the cheapest path possible, perhaps even earning some money along the way.

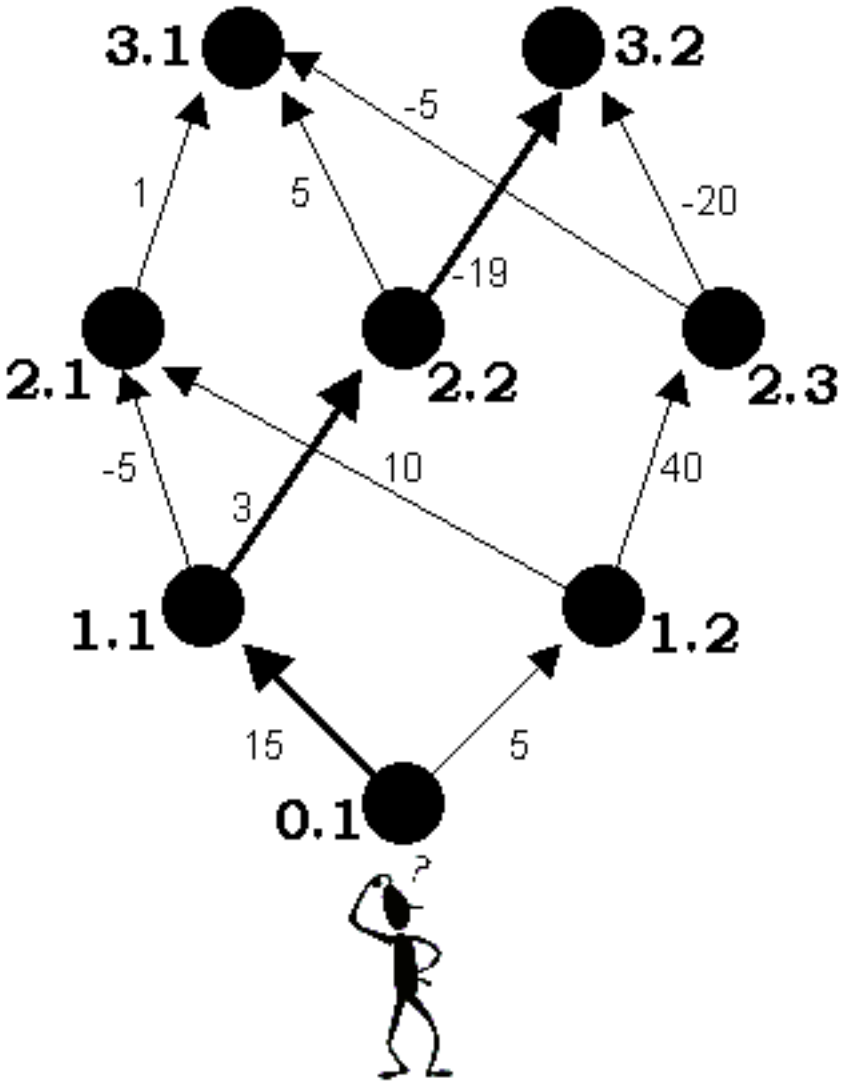
Input

> The first line contains an integer N ($0 < N < 30$) — the number of levels of planets on Ivanushka's map. Following are N blocks of information that describe the interlevel transfers. More precisely, the ith informative block describes the scheme of transfers from (i−1)-st level planets to the ones of ith level. Those blocks are separated with a line that contains only the symbol "*".
>
> Planets of each level are numbered with sequential positive integers starting from 1. Each level contains at most 30 planets. There is only one planet of 0th-level: the one that Ivanushka lives at. The first line of a block contains a number Ki — the number of planets of the ith level. Then follow Ki lines — one for each planet of the ith level. Every line consists of sequences of number pairs - the first number in the pair is the number of a planet of the previous (i-1)st level that has a transfer to the (Ki)th planet in the (i)th level and the second number is the corresponding fee for this transfer. The fee for each transfer is an integer number from −32768 to 32767; a negative fee means that the kind spirit is ready to pay for such a transfer. Each description line is ended by a zero.

Output

> The output should contain only a single integer — the minimum total fees that Ivanushka could pay to travel to any planet of the Nth level. The answer may be negative, which means that Ivanushka will not only get to a heavenly planet, but will earn some galactic bank-notes. It is known that there exists at least one way from Ivanushka's planet to one of the Nth level.

```
In [1]: from IPython.display import Image, display
        display(Image(filename='./week7_hwimg.png'))
```



a) How can we represent the map of transfers between planets as a graph? What are the nodes of the graph? What are the edges of the graph? We talked about several different ways to classify graphs. Classify the graph of transfers between planets according to these distinctions. Is this graph directed or undirected, weighted or unweighted, simple or non-simple, and cyclic or acyclic?

The nodes of the graph will be planets, while the edges of the graph will be transfers between planets. This graph will be directed (transfers are one-way), weighted (transfers cost money to take), simple (no self-loops or multiedges) and acyclic (there's no possible paths from a planet back to itself).

b) One way we might solve this problem is to use the shortest path algorithm (Dijkstra's) we learned about in class today starting from Ivan's home planet to calculate the shortest path to all other planets. How will this algorithm scale based on the number of planets and transfers in the map? How large will this be in the worst possible case?

For the implementation of Dijkstra's algorithm we described in class, its efficiency is $O((E+V)logV)$, where E is the number of transfers and V the number of planets. There are at most 30 levels and 30 planets

per level (and only 1 at level 0), for at most 871 planets. Transfers occur only between planets of neighboring levels, so each level has at most 30^2 transfers from the level before it, for no more than $30 * 30 * 29 = 26,100$ total transfers. Plugging into the formula, that gives a running time of O(182k) operations.

> *c) The graph of transfers has a particularly useful structure - transfers only occur between any level i and the next level $i+1$. Suppose we performed a topological sort of the graph. What would be the first vertex in the sort? How would the levels of the planets be arranged in the topological sort?*

A topological sort arranges the order of all the planets so that when traveling along a transfer between planets we are always traveling from an earlier planet to a later planet. Because transfers only occur from lower to higher level planets, this means that a topological sort would give a sort of planets ordered by level, with Ivan's planet at level 0 at the beginning.

> *d) Suppose that we already know the cheapest way to reach each planet in level $N-1$. Describe a function that takes as input these minimum costs for level $N-1$ and returns the minimum costs to reach each planet in level N. Which transfers do we have to examine to perform this calculation? All of them, or a particular subset?*

Because we know that to reach a planet in level N we must transfer from a planet in level $N-1$, we can use the following algorithm: Initialize an array of minimum costs for level N with very large numbers. Then loop through each transfer from level $N-1$ to level N. The cost to get to this planet in level N using this transfer is the sum of the cost of this transfer and the cost to get to the starting planet in level $N-1$, which we can pull out of the input array. Only transfers from $N-1$ to N must be examined.

> *e) Using the insight from part e, write an $O(V+E)$ dynamic programming solution to the problem, test its correctness on the Timus Online Judge, and attach it to your homework.*

We can solve the problem by starting with a base case of level 0, with a minimum cost array of [0], and then repeatedly calculating the minimum cost array for the next highest level, using the method described in part (d) until we have the final minimum cost array for level N planets. We then simply find the minimum value of these planets and return it.

```
In []: import sys

        N = int(sys.stdin.readline())
        MAXCOST = 1 << 16

        #Initialize our DP array with the cost of the 0th level planet
        mincosts = [0]

        for i in range(N):
            K_i = int(sys.stdin.readline())
            #Initialize the DP array for the next level with large costs
            newcosts = [MAXCOST for k in range(K_i)]
            for k in range(K_i):
                #Read in the transfer pairs (removing the 0 at the end)
                line = [int(i) for i in sys.stdin.readline().split()][:-1]
                #Loop through each pair
                for j in range(len(line) / 2):
                    src, cost = line[j * 2], line[j * 2 + 1]
                    #Update the minimum cost for each planet if this transfer is
                    #cheaper than the current minimum
                    if mincosts[src - 1] + cost < newcosts[k]:
                        newcosts[k] = mincosts[src - 1] + cost
            sys.stdin.readline() #The asterisk lines
```

```
        mincosts = newcosts
    print min(mincosts)
```

Optional No optional work this week.