Problem Set 5

This problem set is due **Tuesday, November 20** at **11:59PM**.

Code for problem 3, as well as a solution template for problem 3 and a LaTeXtemplate for the written problems, have been posted to the course website. Solutions should be turned in at `https://alg.csail.mit.edu` (our submission site).

Programming questions will be graded on a collection of test cases (including example test cases to help you debug). Unless you see an error message, *you will be able to see your grade immediately*. Your grade will be based on the number of test cases for which your algorithm outputs a correct answer within certain time and space bounds. You may submit as many times as you'd like, and only the final submission counts! **Therefore, make sure your final submission is what you want it to be.**

For written questions, full credit will be given only to correct solutions that are described clearly *and concisely*.

Problem 5-1. [20 points] **Graph Transformations**
Consider a directed weighted graph $G = (V, E, W)$ with no negative cycles. Define the set of all weights for all edges $e_{ij} \in E$ as $S = \cup\{w_{ij}\}$.

The following four problems pose transformations from $G = (V, E, W)$ to $G' = (V, E, W')$ which map each weight w_{ij} to a new weight w'_{ij}. Then, true or false: if P is a minimum-weight (s, t)-directed path in G, then P is also a minimum-weight (s, t)-directed path in G'. Provide a brief proof or counterexample for your answer (**three sentences maximum**).

(a) [5 points] Let $w_{min} = min_{i,j}(w_{ij})$, that is, w_{min} is the smallest weight in S (w_{min} may be negative). The transformation is as follows: for all $w_{ij} \in W, w'_{ij} = w_{ij} - w_{min}$.

(b) [5 points] Let k be a constant, positive real number. The transformation is as follows: for all $w_{ij} \in W, w'_{ij} = k \cdot w_{ij}$.

(c) [5 points] $\lfloor x \rfloor$, also known as $floor(x)$, is the largest integer not greater than x. The transformation is as follows: for all $w_{ij} \in W, w'_{ij} = \lfloor w_{ij} \rfloor$.

(d) [5 points] Suppose the weights $w(i, j)$ are all non-negative. The transformation is as follows: for all $w_{ij} \in W, w'_{ij} = w_{ij}^2$.

Problem 5-2. [45 points] **Trip Planning**
Big Bang is planning their Summer 2013 world tour (summer is T days long). There are N cities $c_1, c_2, ..., c_n$ that they are considering to visit. Big Bang begins their tour on day 1 in city c_1, ends their tour on day T in city c_n, and in the meantime, can visit as many or as few cities, in any order, as their schedule permits. Big Bang's algorithms team is in charge of coming up with the schedule.

The algorithms team possesses the list of the M flights between the cities. For each flight, the departure city, arrival city, departure day, arrival day, and flight cost are provided (flights can be arbitrarily long). Assume that the band can arrive at a city, perform, and depart from the city all within the same day (thus all arrivals are in the morning and all departures at night). Furthermore, the band will perform only once per distinct visit. Thus, if the band stays at a city c_k for two weeks, they will only perform once during this time. However, if the band visits c_k, then travels to c_m, and back to c_k, they will have performed three times (twice at c_k and once at c_m).

We've provided an example flight list to help you understand the questions (Table 1). In this example, $T = 7$, $N = 7$, and $M = 18$.

Table 1: Example Flight List

Departure City	Arrival City	Departure Date	Arrival Date	Flight Cost
c_1	c_2	1	2	818
c_1	c_4	3	4	88
c_1	c_7	1	7	0
c_2	c_1	1	2	518
c_2	c_3	2	3	88
c_2	c_5	3	5	114
c_3	c_4	3	4	87
c_3	c_5	6	7	426
c_4	c_1	2	5	89
c_4	c_5	4	5	1212
c_4	c_6	1	5	90
c_5	c_6	5	6	146
c_5	c_7	6	7	107
c_6	c_1	2	5	91
c_6	c_7	6	7	313
c_7	c_1	2	4	407
c_7	c_3	5	6	37
c_7	c_4	6	7	199

(a) [25 points] Big Bang operates on a very tight budget and the band members only require that they be in c_1 on day 1 and c_n on day T.

Come up with an algorithm that will minimize the total cost for Big Bang's tour. Analyze the algorithm's running time and show that it is $O(M + TN)$.

For the example provided on page 3, the solution would be $C_1 \rightarrow C_7$, with cost 0.

(b) [10 points] The truth is that cost and exhaustion is really of no consequence to Big Bang, and so their objective is to maximize the number of times they perform, starting in c_1 on day 1 and ending in c_n on day T.

Come up with an algorithm that will maximize the number of times Big Bang performs. Analyze the algorithm's running time and show that it is $O(M + TN)$. (**Hint**: modify your algorithm from part (a))

For the example provided on page 3, the solution would be $C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4 \rightarrow C_5 \rightarrow C_6 \rightarrow C_7$, with cost 2664.

(c) [10 points] In the problem as specified so far, a flight that departs on day i will arrive at its destination on day $j > i$. As it turns out, there is a special one-time-use portal for Big Bang members which will be open only on day A in city c_p. After entering this portal, they will be taken back in time to the morning of day $B < A$ in city c_q, and the portal will open nevermore. Since everyone in the world already knows about this portal, there aren't any issues with time travel just this once. Furthermore, if Big Bang chooses to exercise this option and performs at city c_r on day k with $B < k < A$, then performs at c_r on day k again after going through the portal, it still counts as two performances.

Solve part (b) again, taking into account the single one-use "time travel" flight. Analyze the algorithm's running time and show that it is $O(M + TN)$. (**Hint**: modify your algorithm from part (b))

Problem 5-3. [40 points] **Land of the Zoombinis: Bubblewonder's Final Abyss**

It turns out that "Bubblewonder Abyss" is a misnomer, as it is actually a long series of abysses. Over the last two weeks, the zoombinis have fallen through countless abysses, and are now at the final abyss. Once they escape, they'll be free! Unfortunately, this abyss is much larger than any they have seen before.

During their journey, the zoombinis have learned a lesson the hard way: extended amounts of sliding is quite painful on the skin. Thus the zoombinis realize that they were silly to minimize the total number of moves; instead, they should have minimized the total distance that they have slid.

Recall that when a zoombini moves in one direction, they will continue moving in that direction until either they encounter an obstruction (i.e. a boulder or another zoombini), in which case they'll stop in the space immediately prior to the obstruction, or until they fall through an exit, in which case they'll escape the abyss and no longer exist on the map.

You must write a function `escape(abyss)`**, which should find a sequence of moves that will allow the zoombinis to escape, and minimizes the sum of the total distances travelled by each of the zoombinis.** We have provided a class representing the state of the zoombinis for you.

You will receive as input an $n \times n$ matrix, representing an abyss, where each cell is one of:

- ' ', an empty space
- 'X', a boulder
- '*', an exit, or
- '0', '1', etc., a number representing some individual zoombini.

In the final abyss that you will be graded on, the Abyss will have size $n = 10$ and there will be $k = 4$ zoombinis. There is roughly 2 boulders per row and column, and 5 exits total, placed randomly. However, we guarantee that it is possible for all the zoombinis to escape. **Don't worry about worst case running time. Just do what you think will pass as quickly as possible!**

Your output should be a list of tuples, (x, d), where x is an integer representing the zoombini being moved, and d is a cardinal directionality, in the set $\{'N', 'E', 'S', 'W'\}$.

To be clear, if we are at position in the matrix with indices (i, j), then north, east, south, and west correspond to $(i-1, j)$, $(i, j+1)$, $(i+1, j)$, and $(i, j-1)$, respectively.

Although the actual abyss is quite large, we can test our algorithm in smaller settings.

Example 1

For example, we might get the abyss:

```
[[' ',' ','0','X','*'],
 [' ',' ',' ',' ',' '],
 ['X','1',' ','X',' '],
 [' ',' ',' ',' ',' '],
 [' ','2','X',' ',' ']]
```

One solution to would be the list,

```
[(2, 'N'), (1, 'E'), (0, 'S'), (1, 'S'), (1, 'E'),
 (0, 'E'), (0, 'N'), (1, 'N'), (2, 'E'), (2, 'N')]
```

which has cost 18.

Example 2

With the abyss:

```
[['X',' ',' ',' ',' ',' ',' ',' ','1'],
 [' ',' ',' ',' ','X',' ','X',' ',' '],
 [' ',' ',' ',' ',' ',' ',' ','2','X'],
 [' ',' ',' ',' ',' ',' ',' ','X',' '],
 ['X',' ',' ',' ','*',' ',' ',' ',' '],
 [' ',' ','X',' ',' ',' ',' ',' ',' '],
 [' ','X',' ',' ',' ',' ',' ',' ',' '],
 [' ',' ',' ',' ','X','0',' ',' ',' '],
 [' ',' ',' ',' ',' ','X','*','X',' ']]
```

One solution would be the list,

```
[(1, 'S'), (1, 'W'), (2, 'W'), (1, 'S'), (2, 'S'),
 (2, 'E'), (1, 'W'), (1, 'S'), (1, 'E'), (0, 'N'),
 (2, 'S'), (1, 'E'), (0, 'W'), (1, 'S')]
```

which has cost 45.