15-451/651 Algorithms, Fall 2018

Homework #4 Due: October 16-19, 2018

This is an oral presentation assignment. There are three regular problems (#1-#3) and one programming problem (#4). You should work in groups of three. The presentation sign-up sheet will be online soon (details on Piazza), and your group should sign up for a 1 hour slot. Each person in the group must be able to present every regular problem. The TA/Professor will select who presents which regular problem. You are not required to hand anything in at your presentation, but you may if you choose. The dates of the orals will be October 16th through 18th.

The programming problem will be submitted to autolab with the name beer.*. It is due at 11:59pm on Friday, October 19. You will not have to present anything orally for the programming problem. Please include a comment in your program explaining the algorithm you used. You can discuss the problem with your group-mates, but must write the program by yourself. Do not employ any code that you did not write yourself.

1. **Knights Who Stay Glued**

   You are given an \( n \times n \) chessboard with \( k \) knights (of the same color) on it. Someone has spilled superglue on \( k \) of the squares, and if a knight ever finishes his move on one of these glue squares, it becomes stuck forever.

   Additionally (and this is why we can’t have nice things) someone has cut out some of the squares so the chessboard has holes in it.

   You are given an initial position of the knights. The knights move as they do in regular chess, but unlike regular chess, on each turn all the knights move at once (except of course the stuck ones). At the end of each move, a square cannot be occupied by more than 1 knight. Hole squares can’t be occupied by knights either (but they do count as squares that the knight can jump over).

   Give an \( O(t \times \text{poly}(n)) \)-time algorithm to determine whether you can use \( \leq t \) moves to move all the knights from their initial positions to new positions where they are each stuck at a glue square.

2. **A Game**

   Bryan and Katrina play the following game. Bryan has three cards. Card A has 0 on top and 5 on the bottom. Card B has 7 on top and 3 on the bottom, and card C has 10 on top and 0 on the bottom. Simultaneously Bryan picks a card while Katrina selects “top” or “bottom.” Katrina pays Bryan the number on the selected side of the selected card.

   (a) This is a two player zero sum game with discrete choices for each player. Write the pay-off matrix where Bryan is the row player and Katrina is the column player.

   (b) What is Katrina’s optimal mixed strategy?

   (c) What is Bryan’s optimal mixed strategy?

   (d) What is the value of the game (to Bryan)?
3. Constrained Profits

In the constrained profits problem you have \( n \) possible tasks \( v_1, \ldots, v_n \) that you can perform. Each task \( v_i \) has a corresponding net profit \( p_i \), which may be positive or negative.

You are also given a directed graph \( G = (V, E) \) of constraints between these tasks, where edge \((v_i, v_j)\) means that if task \( v_i \) is performed, then \( v_j \) must also be done. In other words, to do task \( v_i \) and earn or pay the profit \( p_i \), you must also do task \( v_j \) and earn or pay its profit \( p_j \).

The problem is to decide which of the tasks to do so as to maximize the profit. So one way to look at this is that you have to label each vertex with \( T \) or \( F \). (\( T \) means that the task is done, \( F \) means that it is not done.) It must be the case that there is no edge from a vertex labeled \( T \) to one labeled \( F \). The profit of a solution is simply the sum of the \( p_i \) values over all vertices labeled \( T \).

(a) Let the vertices of \( G \) be divided into the profitable vertices \( P \) (where \( p_i \geq 0 \)) and the costly vertices \( C \) (where \( p_i < 0 \)). So \( P \) and \( C \) are disjoint and \( V = P \cup C \).

For this part, suppose that the graph \( G \) is bipartite, and all of the edges of the graph are directed from \( P \) to \( C \).

Show how to model this problem as a flow graph \( H \), and prove that your method correctly models the problem.

Here are some more hints for how to approach this. Set up a flow graph \( H \) such that there’s a one-to-one correspondence between cuts in \( H \) (of finite value) and valid labelings of \( G \) with \( T \) and \( F \). The cost of the cut in \( H \) should be related to the profit obtained in \( G \), such that the lower the cost of the cut the higher the profit. Explicitly state this relationship.

The min cut in this graph should allow you to derive (using your one-to-one correspondence) the optimal solution to this constrained profits problem. (Working out some simple examples will be helpful.)

(b) Solve (with proof, and with a way to construct an optimal solution) the fully general constrained profits problem as described before part (a).
4. **Bart Likes Beer**

Depending on how many he’s had, Bart spells the name of his favorite beverage “ber”, “beer”, “beeer”, “beeeer” etc. In the language of regular expressions, a string like this is denoted “be+r”.

In this programming problem, you’re given a rectangular board of letters. Bart wants to find as many be+r’s as possible in this matrix. That word appears in the board if there is a sequence of squares such that “b” is in the first one, “e” is in the second one, and subsequent ones, finally ending in an “r”. Each letter is next to (either horizontally or vertically) the previous one. You are allowed to use a square at most once.

The *score* of a board is the maximum number of times you can simultaneously find be+r in it. In other words, find one be+r and cross off those cells (they can’t be used again). Then repeat this step as many times as possible. (See the examples below.)

The time limit is 10 seconds.

**Input:**

The first line contains two space-separated integers: r and c. The next r lines are strings of c characters from the set {“b”, “e”, “r”, “.”} followed by a newline. 1 ≤ r, 1 ≤ c, and r · c ≤ 100 000.

**Output:**

Print the maximum number of disjoint be+r’s that can be simultaneously found on the board (the score described above), followed by the word “beer” or “beers”, whichever is appropriate.

Here are some sample input and output pairs:

```plaintext
1 12
rebereberebe
3 beers

1 12
berrebergerb
4 beers

5 4
b..b
eeee
.e..
eeee
r..r
1 beer

4 4
beer
e..b
erb
rbbb
1 beer

1 1
b
0 beers
```