15-451/651 Algorithms, Spring 2019

Homework #4 **Due: Monday–Thursday, March 4–7, 2019**

This is an oral presentation assignment. Same rules as HW2. Prove your answers correct for problems 1-3. **Note that the presentation days are Monday–Thursday, because Friday is mid-semester break.**

We are allowing you to submit the programming question on the Sunday at the end of spring break (March 17th 11:59pm). *That being said, the course staff will be taking a much-needed break during that time too, so there are no guarantees that your questions will be answered over the break — please plan accordingly.*

(25 pts) 1. **(Fall Foliage in the Spring)**

Let $G = (V, E)$ be an undirected graph with the vertex set V and edge set E. Each edge has an integer weight $w_e > 0$. A $(k, n-k)$-*partition* of G is a coloring of the vertices with two colors (red/blue) such that there are k red nodes, and $n-k$ blue nodes.

The goal is to find a $(k, n-k)$-partition of G where the total weight of *split* edges (edges with one endpoint red and the other blue) is minimized. In general there is no known polynomial time algorithm for this problem. However, if the graph is a tree, the problem becomes easier.

Design a polynomial-time algorithm to solve this problem when the input graph G is a *binary tree*. Formally, a binary tree (for our purposes) is a tree rooted at some node r, and each node has at most two children. Prove the correctness of your algorithm and analyze its running time. For full credit, your algorithm should take $O(n^3)$ time.

(25 pts) 2. **(Sort of Magic Square)**

Consider the Generalized Magic Square problem: Given k, and two lists of n non-negative integers $\vec{r} = (r_1, \ldots, r_n)$ and $\vec{c} = (c_1, \ldots, c_n)$, we ask whether there is an $n \times n$ grid of integers from the set $\{0, 1, \ldots, k\}$ such that row i sums to r_i and column j sums to c_j. We assume $\sum_i r_i = \sum_j c_j$.

Examples: With $k = 1$ and $n = 3$ with $\vec{c} = (1, 2, 0)$ and $\vec{r} = (1, 1, 1)$ (answer=yes) or $\vec{r} = (3, 0, 0)$ (answer=no):

Yes

1	0	0	1
0	1	0	1
0	1	0	1
1	2	0	

No

1	1	1	3
			0
			0
1	2	0	

Use network flow to create a polynomial-time algorithm to decide whether it is possible to design a grid containing integers from 0 to k that obeys the given $\vec{r}$ and $\vec{c}$ sums. Illustrate your construction with the "yes" example above.

(25 pts) 3. **(You Think You've Got Problems?)** You just realize that the semester is almost half-over, and you need to get your act together. Your textbook has n chapters, each quite technical, where chapter i costs C_i dollars to read. ("Time is money," as they say.)

The homework consists of a set of m problems $p_1, p_2, \ldots, p_m$. For each problem p_j, there is a subset $P_j \subseteq \{1, \ldots, n\}$ of the chapters that it depends on. If you have read *all* the chapters in P_j you can solve the problem p_j, but if you've missed even one of the chapters in P_j you cannot solve the problem. Solving p_j gives you V_j dollars of value. The *net utility* is the value of the problems solved minus the cost of the chapters read. The goal is to find a subset $R \subseteq \{1, \ldots, n\}$ of the chapters to read, to maximize the net utility you get.

E.g., if $P_1 = \{2, 3, 5\}$, $P_2 = \{1, 2, 3\}$ and $P_3 = \{2, 3, 4\}$. Suppose the costs for the $n = 5$ chapters are $1, 4, 3, 8, 1$ respectively, and values for the three problems are $V_1 = 14, V_2 = 4, V_3 = 7$. If you have read chapters $\{2, 3, 4, 5\}$ then your cost is $4+3+8+1 = 16$ and the value you get from having solved P_1 and P_3 is $14+7 = 21$. So the net utility is $21-16 = 5$. On the other hand, having read just $2, 3, 5$ you'd have net utility $14 - (4 + 3 + 1) = 6$. And having read just $1, 2$ you'd get net utility $0 - (1 + 4) = -5$.

Show how to use an s-t-min-cut algorithm to solve this problem in polynomial time.

Hint: Can you solve the problem of *minimizing* the cost of the chapters you read *plus* the sum of the values of problems you *did not* solve. Why is solving this problem this useful? Think about how you could solve this problem using an s-t min-cut algorithm.

(25 pts) 4. **(Delightful Dominos.)** In this programming problem you're given a rectangular board of square cells. Some of the cells are blocked. You are to compute the maximum number of dominos (1×2 tiles) that can be placed on the board. Each domino will be in either a vertical or horizontal orientation, and is placed on two neighboring squares, neither of which must be blocked. The time limit is 10 seconds. The program should be called `dominos.c`, or the analogous name based on what language you use.

Input: The first line contains two space-separated integers: r and c. The next r lines are strings of length c comprised of the characters "`.`" and "`x`". The "`x`" characters denote cells of the board that are blocked. $1 \le r, c \le 30$.

Output: Print the maximum number of dominos that can be placed on the board satisfying these constraints. The format of the output is illustrated below.

Input 1:

```
2 3
...
...
```

Output 1:

```
3 dominos
```

Input 2:

```
5 4
.xx.
xx.x
x...
xx.x
.xx.
```

Output 2:

```
1 domino
```

Input 3:

```
2 5
..xx.
.....
```

Output 3:

```
4 dominos
```