CSE421: Design and Analysis of Algorithms May 13, 2020

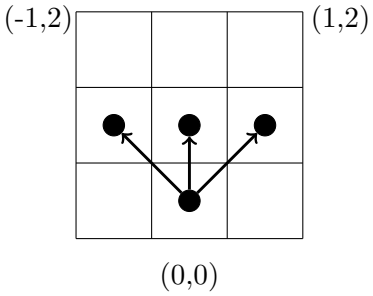
Homework 6

Shayan Oveis Gharan Due: May 20, 2021 at 23:59 PM

P1) Drogo, Daenerys's Dragon started practicing free climbing. The wall it is practicing on is $n+1$ squares tall and $2k+1$ squares wide. The bottom left square is at $(-k, 0)$ and the top right is at (k, n)

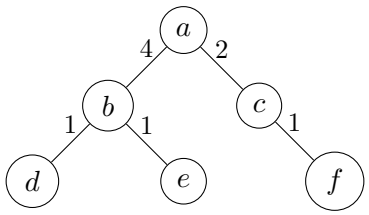
Little Drogo starts from the block $(0, 0)$. At each step Little Drogo climbs one block and either moves left one block, right one block or stays in the same line of climbing. So from (a, b), Little Drogo can go to $(a-1, b+1)$, $(a, b+1)$ and $(a+1, b+1)$, as long as the destination square exists. So, for example, from $(0, 0)$ it can go to $(-1, 1), (0, 1)$, or $(1, 1)$ (see below).

Design a polynomial time algorithm that outputs in how many ways can little Drogo climb to the top level? For example, if $n = 2, k = 1$ (as in the case below) the answer is 7.



P2) Stock market gets real this time. Recall that you are so lucky to have a friend who knows the price of a specific commodity in the next n days in the stock market; say the price will be p_i in day i and it is given to you in the input. Every day you can either buy the commodity (if you don't have it already) or sell it (if you have it). In this case, each time that you **sell** you have to **pay** c **dollars**, for a given integer $c > 0$ in the input. As before, there is a single commodity in this market and you cannot buy two copies of it. Design an $O(n)$-time algorithm which outputs the maximum possible gain. If your code runs in polynomial time as opposed to $O(n)$ you will receive 18 points out of 20.

P3) You are given a tree T where every edge e has weight $w_e \geq 0$. Design a polynomial time algorithm to find the weight of maximum weight matching in T. Remember that a matching M is a set of edges of T such that any vertex of T is incident to *at most* one edge of M.



For example, in the (above) tree, the maximum weight matching has edges $(a, b), (c, f)$ with weight $4 + 1 = 5$.

P4) Interstate highway 5 is a straight highway from Washington all the way to California. There are n villages alongside this highway. Think about the highway as an integer axis, and the position of village i is an integer x_i along this axis. Assume that there are no two villages in the same position, i.e., $x_i \neq x_j$ for $i \neq j$. The distance between two villages x_i, x_j is simply $|x_i - x_j|$.

USPS is interested in building k post offices in some, but not necessarily all of the villages along highway 5, for some $1 \leq k \leq n$. A village and the post office in it have the same position. We want to choose the positions of these post offices so that the *sum* of the distances from each village to its *nearest* post office is minimized. Design an algorithm that runs in time polynomial in n and outputs the minimum possible sum of distances to the optimal location for post offices. For example, given the following location of 5 cities if $k = 2$ then the optimal location for post offices are at village 0 and 4 and your algorithm should output 3 as the *sum* of distance to nearest post offices.

P5) **Extra Credit:** Given a sequence of positive numbers $x_1, \ldots, x_n$ and an integer k, design a polynomial time algorithm that outputs

$$\sum_{S \in \binom{n}{k}} \prod_{i \in S} x_i,$$

where the sum is over all subsets of size k.