# HW 7

Due on **Wednesday, October 23, 2024 at 9am CST**

**CS/ECE 374A: Intro. Algorithms & Models of Computation, Fall 2024** Version: **1.02**

**Submission instructions as in previous homeworks.**

If you describe a DP solution for any of the questions below, follow the guidelines from the previous homework.

**13** (100 PTS.) Elections I.

In the United States of the Grid, the states are organized in a $[\![n]\!] \times [\![n]\!]$ grid, the elections are fair, and everyone is slightly above average, where $[\![n]\!] = \{1, \ldots, n\}$. Here a state $(i, j)$ has $v(i, j) \in [\![54]\!]$ electoral votes, and campaigning in the state costs $p(i, j) \in [\![400]\!]$ (say in millions of dollars).

You are given a budget $\beta \leq 800n$, and you start your campaign at $(n, n)$ and ends it at $(1, 1)$. When in a state $(i, j)$, you can either campaign in it (and pay $p(i, j)$), or alternatively skip it (and pay nothing, how great is that?). After being at state $(i, j)$, you must move to either $(i, j - 1)$ or $(i - 1, j)$. If you campaign in a state $(i, j)$, you get all its electoral vote $v(i, j)$, otherwise, you get nada.

Given as input the above information (i.e., the numbers $n$ and $\beta$, and two matrices $v[n, n]$ and $p[n, n]$), describe an algorithm, as fast as possible, that wins as many electoral votes as possible overall, while not exceeding the budget $\beta$. (What is the running time of your algorithm?)

The algorithm should output the path, and the states along it one should campaign in.

**14** (100 PTS.) Make span for $k$ robots great again.

You are given $k$ robots. Each robot is a disk of radius 1 in the plane. The $i$th robot is given a sequence of $n$ locations it needs to travel through $p_i(1), \ldots, p_i(n)$. The robots all start at their start locations $p_1(1), \ldots, p_k(1) \in \mathbb{R}^2$. At the start of a minute, all the robots must be in one of their valid locations (say, the $i$th robot is at $p_i(\ell(t))$ at time $t$), and during this minute each robot can either stay put, or travel to its next location (e.g., $p_i(\ell(t) + 1)$). Several robots might move in the same time. A ***configuration*** is a tuple of $k$ integers $(i_1, i_2, \ldots, i_k)$ specifying the locations of the $k$ robots at the beginning of a minute. Such a configuration is ***feasible***, if for all $\alpha < \beta$, we have $||p_\alpha(i_\alpha) - p_\beta(i_\beta)|| \geq 2$ (i.e., no two robots collide).

The task at hand is to start from configuration $(1, 1, \ldots, 1)$, and arrive to $(n, n, \ldots, n)$ in such a way that all the configurations used in the motion are feasible, two consecutive configurations in the motion represent a valid motion that can carried out in one minute, and the total span (i.e., total time) spent is minimized. For simplicity, you can assume that the naive schedule where the first robot first follow its path, then the second robot follow its path, etc, is feasible (but its span is "bad" - it is $k(n - 1)$.)

Describe an algorithm, as fast as possible, to solve this problem and output the optimal path. What is the running time of your algorithm as a function of $k$ and $n$?