Shortest Paths
and Dynamic Programming
Problem Set 4 – CS6515 (Spring 2025)

- This problem set is due on **Thursday February 13th**.
- Submission is via Gradescope.
- Your solution must be a typed pdf (e.g., via LaTeX, Markdown, etc. Anything that allows you to type math notation) – no handwritten solutions.
- Please try to make your solutions as concise and readable as possible. Most problems will have solutions that are no more than a page long. Consider using bullet points and adding space to break up large paragraphs into smaller chunks.
- There are 3 problems. Each problem is graded with 20p. **There is +1p bonus per problem** for stating (i) how long it took you to solve that problem, and (ii) how long it took you to type the answer.

Remark on algorithm descriptions To make things easier for the TAs to grade, please use a combination of plain English and mathematical notation. Do not write code (C, Java, etc.). For example, you can say something like "$x := \max_{i=0,\ldots,n-1}$ `A[i]`" or "Find the maximum value in the array A" instead of writing a for-loop that computes the maximum.

10 Satisfying Inequalities

We have n variables $x_1, x_2, ..., x_n \in \mathbb{R}$ and are given m inequalities of the form $x_i - x_j \leq c_{i,j}$. These are given as arrays $C[1...m]$, $I = [1...m]$, $J = [1...m]$ which represents $x_{I[k]} - x_{J[k]} \leq C[k]$. Here $C[k] \in \mathbb{R}$ and could be negative.

1. Give an algorithm that, given C, I, J, returns YES/NO depending on whether one can find values for $x_1, ..., x_n$ that satisfy all inequalities.
2. Prove that your algorithm is correct.
3. What is your algorithms time complexity?

For full points, your algorithm's time complexity should be $O(mn)$.

Example $x_1 - x_2 \leq -100$, $x_2 - x_3 \leq 10$, $x_3 - x_1 \leq 20$. Here we cannot find a solution because

$$0 = (x_1 - x_2) + (x_2 - x_3) + (x_3 - x_1) \leq -100 + 10 + 20 = -70.$$

Another example: $x_1 - x_2 \leq -100$, $x_2 - x_3 \leq 10$, $x_1 - x_3 \leq 20$. Here $x_3 = 0, x_2 = 10, x_1 = -90$ is a valid solution.

Hint: Can you phrase this problem as a graph problem? Why are they equivalent? You'll need to prove both directions (if the graph... then there is no solution. If the graph not... then the a valid solution is ...).

2nd Hint: If you want your graph to have a vertex that reaches every other vertex, what about simply creating such a vertex? Does adding a new vertex break anything?

11 Building Items

We have 2 kinds of resources (eg wood and stone). Let w and $s \in \mathbb{N}$ be the amount we have of each of the two resources.

We also have n templates that describe items we can build. Let $W[1...n], S[1...n]$ be arrays where $W[i]$, $S[i]$ describes the cost it takes to build an item of type i. We can build multiple of each item! Array $V[1...n]$ describes the amount of value for item i.

We want to design an algorithm that receives $W[1...n], S[1...n], V[1...n], s, w$ as input, and then returns the maximum value we can build.

Problem Construct an algorithm by answering the following questions:

1. Give the array and describe what an entry means. (E.g., $T[i, j]$ or $T[i, j, k]$ etc is...)
2. Formulate the recursion as compact as possible with mathematical notation using your array. Explain briefly why it's correct.
3. Which cells of the array form the base case, and what are their initial values?
4. Which cell contains the solution?
5. What is the time complexity to fill up the table with values?

For full points, your algorithm should be as fast as possible.

12 A* Correctness

Let h be a consistent heuristic. Consider A* with visited check. We want to prove it produces the correct distance from start vertex s to target vertex t. You are allowed to prove it any way you like, but the following proof by contradiction may be easiest in my opinion.

Proof suggestion: Assume we run A* and at some point it visits a vertex v and assigns some value to `distance[v]` that is incorrect, i.e., not the correct distance from s to v. We want to argue there is a contradiction, i.e., that this cannot have happened.

Let us assume v is the first vertex where it assigns a wrong value and let us pause the algorithm in exactly the moment where it assigns an incorrect value to `distance[v]`. Show the following:

- Show $\mathit{priority}(v) > \text{dist}(s, v) + h(v)$.
- Let P be the correct shortest path from s to v, and let w be the first vertex on that path that was not visited yet by A*. Show $\mathit{priority}(w) \leq \text{dist}(s, v) + h(v)$.
- Argue why $\mathit{priority}(w) \geq \mathit{priority}(v)$ and how that leads to a contradiction.

Here $\text{dist}(\cdot, \cdot)$ refers to the correct distance, i.e., length of the shortest paths.