Precept Outline

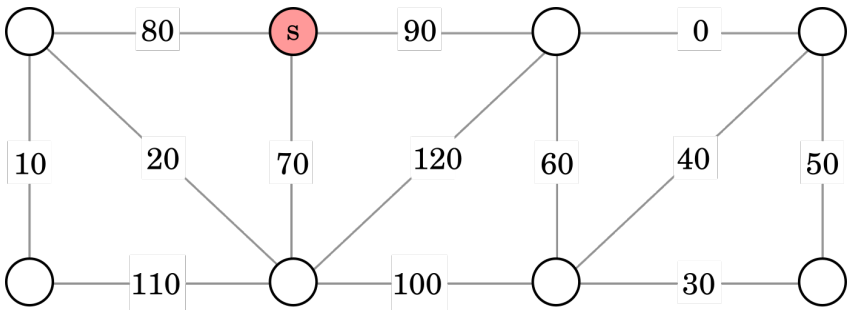
- Review of Lectures 16, 17 and 18:
  - Minimum Spanning Trees
  - Shortest Paths
  - Dynamic Programming

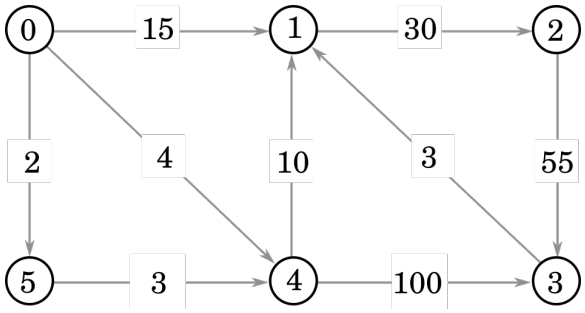
Relevant Book Sections

- Book chapters: 4.3 and 4.4

A. Review: MSTs, Shortest Paths and Dynamic Programming

Your preceptor will briefly review key points of this week's lectures. They may use the following graphs to trace examples:





B. Dorm Rooms and Routers

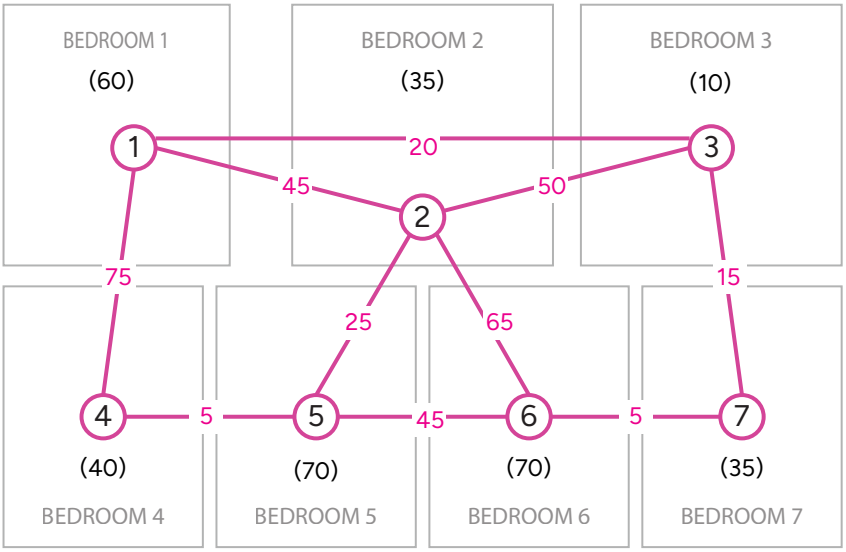
A college has just unveiled a brand-new dorm facility with n rooms. They need to make sure all of them have an internet connection (of course), and are looking for the most cost-effective way to do so. Room number i has internet access if either of the following is true:

- There is a router installed in room i.
- Room i is connected by a fiber path to a room j which has internet access.

Installing a router in room i costs $r_i > 0$, and putting down fiber between rooms i and j costs $f_{ij} > 0$.

The goal of this problem is to determine in which rooms to install a router and which pairs of rooms to connect together with fiber, so as to minimize the total cost.

Formulate this as a *minimum spanning tree problem*. Start by defining a graph $G = (V, E)$ with vertices $V = \{1, 2, \ldots, n\}$ and edges/edge weights that depend on r_i and f_{ij} corresponding to the diagram below. Then define a modified graph G' on which you can run an existing MST algorithm (Kruskal's or Prim's).

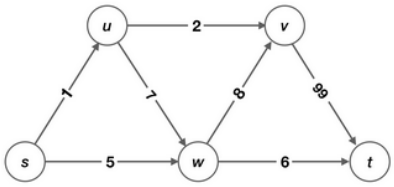


This instance contains 7 dorm rooms and 10 possible connections. The router installation costs are indicated in bold and parentheses; the fiber costs are given on the edges.

C. Shortest Teleport Path (Fall'14 Final)

Given an edge-weighted digraph G with non-negative edge weights, a source vertex s and a target vertex t, find a shortest path from s to t where you are permitted to *teleport* across one edge for free. That is, the weight of a path is the sum of the weights of all but the largest edge weights in the path.

For example, in the edge-weighted digraph below, the shortest path from s to t is $s \to w \to t$ (with weight 11) but the shortest teleport path is $s \to u \to v \to t$ (with weight $1 + 2 + 0 = 3$).



A full solution should run in $O(E \log V)$ time and $O(V + E)$ extra space.

D. Dynamic Programming

This question was adapted from the Fall'23 Final Exam.

Suppose you are taking an idealized exam with n questions and have m minutes to complete it. Question j is worth p_j points and takes t_j minutes to earn the points. Your goal is to maximize the number of points earned in the allotted time. Assume that all p_j and t_j are positive integers (and that there is no partial credit in this idealized exam).

You will solve this problem using dynamic programming. Let's start by defining some meaningful overlapping subproblems. For each i and j with $0 \leq i \leq m$ and $0 \leq j \leq n$, give a definition of OPT(i, j) that you think might be useful.

OPT$(i, j) =$

Given this definition, fill in the following blanks:

OPT$(0, j) =$

OPT$(i, 0) =$

OPT$(m, 1) =$

Recall that your goal is to maximize the number of points earned in the allotted time. Give the solution to the problem in terms of some value of OPT(i, j)

Now we need to compute the value of OPT(i, j) in order to solve the problem. Define a recurrence relation to find OPT(i, j) given smaller values of i, j.

$$\text{OPT}(i,j) = \begin{cases} & \text{if} \\ & \text{if} \end{cases}$$

Now, put everything together. Fill in the following Java code to solve this problem.

```
int maximumExamScore(int m, int n, int[] points, int[] times) {
    int[][] opt = new int[m+1][n+1];
    for (int i = 1; i <= m; i++) {
        for (int j = 1; j <= n; j++) {
            if (times[j] > i)
                opt[i][j] = ____________________________________________________;
            else
                opt[i][j] = ____________________________________________________;
        }
    }
    return opt[m][n];
```

```
12 }
```

Finally, what is the running time of this algorithm?

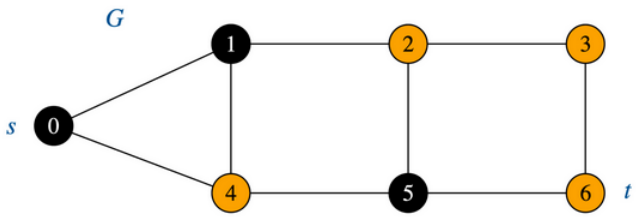
E. Optional bonus problems

Part 1: Shortest Tiger Path (Spring '23 Final)

Consider a graph G in which each vertex is colored black or orange. A *tiger path* is a path that contains exactly one edge whose endpoints have opposite colors.

Our goal is to solve the *shortest tiger path problem*: given an undirected unweighted graph G and two vertices s and t, find a tiger path between s and t that uses the fewest edges (or report that no such path exists).

For example, the shortest path between $s = 0$ and $t = 6$ in the graph below is $0 \to 4 \to 5 \to 6$, but it is not a tiger path; the shortest tiger path is $0 \to 1 \to 2 \to 3 \to 6$.



Formulate the shortest tiger path problem as a traditional (unweighted) shortest path problem in a directed graph. Specifically, define a digraph G', source s, and destination t such that the length of the shortest path from s to t in G' is always equal to the length of the shortest tiger path between s and t in G. For simplicity, you may assume that s is black and t is orange.

For full credit, the number of vertices in G' must be $\Theta(V)$ and the number of edges must be $\Theta(E)$, where V and E' are the number of vertices and edges in G', respectively.