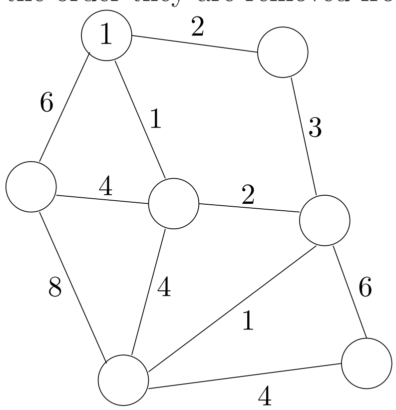
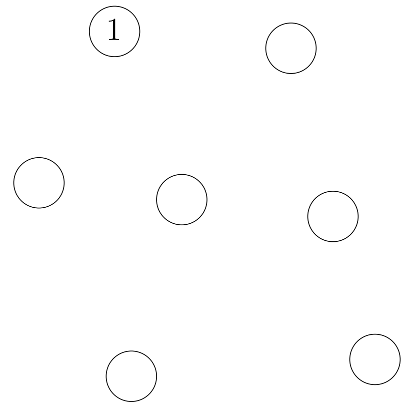
1. [4 marks] Use the integral method to give a good upper bound on $\sum_{i=1}^{n^2} \frac{1}{2i+3}$. Use big-O notation.

2. [4 marks] Suppose we run Dijkstra's algorithm on the undirected graph shown on the left, using the vertex labelled 1 as the root. On the right, draw the shortest path tree that is computed. Number the vertices in the order they are removed from the priority queue, starting with vertex 1.





3. [10 marks] Let $T[1..m, 1..n]$ be a Young tableau. (Recall that this means each row is sorted and each column is sorted.) The following algorithm searches for a location in T that contains the integer k.

SEARCH($T[1..m, 1..n], k$)
    % Preconditions: T is a Young tableau. Every element in column n is ∞. Every element in row m is ∞.
    $i \leftarrow 1$
    $j \leftarrow n$
    loop
        % invariant: $1 \leq i \leq m$ and $1 \leq j \leq n$ and $T[i, j] \geq k$ and
            k does not appear in the first $i - 1$ rows of T (i.e. in $T[1..i - 1, 1..n]$)
        exit when $j = 1$ or $i = m$ or $T[i, j] = k$
        if $T[i, j - 1] \geq k$ then $j \leftarrow j - 1$
        else $i \leftarrow i + 1$
    end loop
    if $T[i, j] = k$ then output (i, j)
    else output "k does not appear in T"
end SEARCH

[6] **(a)** Prove the loop invariant holds at the start of every iteration of the loop.

[3] **(b)** Suppose the loop terminates with $j = 1$. Explain why the algorithm outputs the right answer.

[1] **(c)** Give the (worst-case) running time of the algorithm in terms of m and n. Use Θ notation. Do not prove your answer is correct.

4. [5 marks] Prim's algorithm uses a priority queue. Instead of using a heap to implement the priority queue, we could simply use an array $P[1..n]$, where $P[i]$ stores the priority of the ith vertex.

[3] **(a)** If we use this implementation, what would the running time of Prim's algorithm be for a graph with n vertices and m edges. Use Θ notation. (Assume the input graph is given using the adjacency list representation.) *Briefly* justify your answer.

[2] **(b)** For what type of graph would this implementation be better than the implementation that uses a heap? *Briefly* justify your answer.

5. [8 marks] Given an unsorted array of n distinct elements, we wish to output the k smallest elements in sorted order (where $1 \leq k \leq n$). Use Θ notation to state worst-case running times (in terms of n and k) as simply as possible.

[2] **(a)** We could sort all n elements and list the first k. Which sorting algorithm should we use? State the running time of this solution.

[1] **(b)** We could make the array into a heap (with each element being smaller than its children) using the HEAPIFY procedure. We could then extract the minimum element k times. What is the running time of this solution?

[3] **(c)** Describe, in one or two sentences, an algorithm that solves the problem in $O(n + k \log k)$ time.

[2] **(d)** Answer part (a) again, this time assuming that the elements in the input array all come from the set $\{1, 2, \ldots, 3n\}$.

6. [8 marks] Let G be a directed, acyclic graph with n vertices and m edges. Assume the vertices $v_1, v_2, \ldots, v_n$ of G have already been topologically sorted and that you are given the adjacency list representation of G. Each edge (v_i, v_j) has a bandwidth $b(v_i, v_j) > 0$. The bandwidth of a path is the *minimum* bandwidth of any edge along that path.

For $1 \leq i \leq j \leq n$, let $P[i, j]$ be the *maximum* bandwidth of any path from v_i to v_j. ($P[i, j] = 0$ if there is no such path.)

[6] **(a)** Give a dynamic programming algorithm that efficiently computes all elements of the array P.

[2] **(b)** What is the (worst-case) running time of your algorithm. State your answer in terms of m and n using Θ notation, and *briefly* justify your answer.

7. [10 marks] Three people are working in a workshop. There are n jobs. Each job will take one hour of work by one person to complete. Job i has deadline d_i, where $1 \leq d_i \leq n$. If job i is completed before d_i, it will generate p_i dollars of profit. If it is completed after d_i, it generates no profit. We want to choose some jobs to assign to timeslots of the three workers so that the total profit is maximized.

[3] **(a)** A greedy algorithm can solve this problem by looking at the jobs one by one (in some order) and deciding what to do with each one. Give such an algorithm.

[3] **(b)** After your algorithm has looked at i jobs, it has constructed a partial solution Q_i. Define Q_i and Q^* as mathematical objects and define, formally, what it means for an optimal solution Q^* to extend Q_i.

[4] **(c)** Let $1 \leq i \leq n$. Assume there is an optimal solution Q^* that extends Q_{i-1}. Show that there is an optimal solution $\hat{Q}$ that extends Q_i, in the case where job i is done at the same time in Q_i and Q^*, but by different workers.

8. [1 mark + 5 bonus marks] There is a large city on an island. The Easter Bunny has left 100 Easter eggs at various intersections in the city. Assume you have no map of the city (and no paper on which to draw one). However, you have a big bag full of flat pebbles which you can leave at places in the city as markers. Each pebble has an arrow painted on one side. Assume you have enough pebbles so that you could leave one at every intersection in the city, plus one extra pebble to mark your starting point.

[1] **(a)** What is the name of the algorithm you should use to find all the Easter eggs (efficiently)?

[5 bonus] **(b)** **This is a bonus question. Answer it only if you have extra time.**
Explain how to implement the algorithm using the pebbles. (I.e. give a detailed algorithm describing what you would do whenever you reach an intersection.) Do *not* prove your answer is correct.