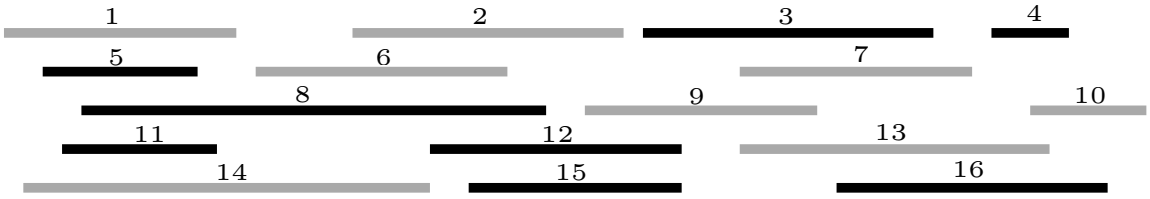
CS:3330 Exam 2, Spring 2017

Tuesday, April 4 2017, 6:30 pm to 8:30 pm

---

1. Let X be a set of n intervals on the real line. A subset of intervals $Y \subseteq X$ is called a *tiling path* if the intervals in Y cover the intervals in X, that is, any real value that is contained in some interval in X is also contained in some interval in Y . The *size* of a tiling path is just the number of intervals in the tiling path.



In the above example, you are given an input with 16 intervals and the intervals shown in gray form a tiling path. This is because every point (on the real line) contained in any interval (black or gray) is contained in a gray interval. However, this tiling path with 8 intervals is not the tiling path with smallest size.

   (a) Write down a smallest size tiling path for the above example. Use the provided labels 1 through 16 to present your answer.

   (b) Describe a polynomial-time *greedy* algorithm to compute the smallest tiling path of X. Your input consists of an array $I[1, \ldots, n]$, where for each j, $1 \leq j \leq n$, $I[j]$ represents an interval (s_j, f_j) with start time s_j and finish time f_j. You can write your answer in pseudocode or plain English. In either case, your algorithm should be super-clear. Since this is a greedy algorithm, it should be fairly easy to describe.

   (c) Prove that your algorithm in (b) is correct. You might want to mimic the "exchange argument" that was used in the proof of correctness of the "Earliest Finish Time First" algorithm for the Interval Scheduling problem.

2. This problem is on Huffman codes.

   (a) What is the optimal binary Huffman encoding for the following set of frequencies, based on the first 8 Fibonacci numbers?

$$a : 1,\ b : 1,\ c : 2,\ d : 3,\ e : 5,\ f : 8,\ g : 13,\ h : 21.$$

   Your answer should consist of a binary string code for each letter "a" through "h."

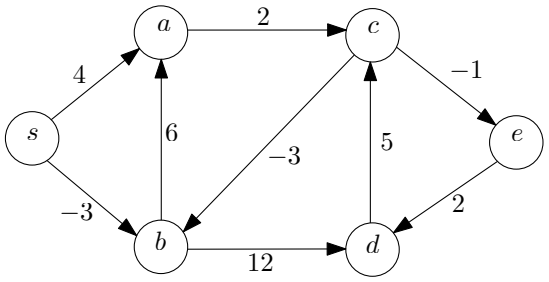
   (b) Suppose that you are given n letters whose frequencies are the first n Fibonacci numbers. Generalizing your answer in (a), write down the encoding of the two letters with frequency 1, in the optimal binary Huffman encoding.

   (c) The Huffman encoding algorithm naturally generalizes to ternary codewords, i.e., codewords that use symbols 0, 1, and 2. Given frequencies:

$$a : 45,\ b : 13,\ c : 12,\ d : 16,\ e : 9,\ f : 5$$

   find an optimal *ternary* Huffman encoding for these letters.

3. Execute the Dantzig-Ford algorithm (version 2) on the graph shown below.



Assume that the "bag" data structure has been implemented as a stack ("Last In First Out" data structure.) For every iteration of the **while**-loop show:

(a) the contents of the stack at the start of the **while**-loop iteration,

(b) the vertex that is taken out of the stack during the **while**-loop iteration,

(c) the edges that are relaxed in the **while**-loop iteration, and

(d) the new $dist(\cdot)$ values that result due the relax operations in (c).

After writing down the information mentioned above, write down all of the *final* $dist(\cdot)$ and $pred(\cdot)$ values after the algorithm completes execution.

4. When there is more than one shortest path from one node s to another node t, it is often convenient to choose a shortest path with the fewest edges; call this the *best path* from s to t. Suppose we are given a directed graph G with positive edge weights and a source vertex s in G. Describe an algorithm to compute best paths in G from s to every other vertex. Your algorithm should be a simple modification to Dijkstra's algorithm; you just have to tell us the modification and this may take no more than 3-4 sentences. You do not have to prove correctness or analyze the running time.
(**Note:** If this problem seems familiar, it is because you have already seen it in a homework.)

5. Suppose that we are given a connected, edge-weighted graph $G = (V, E)$ in which all edge weights are in the range $\{1, 2, \ldots, 10\}$. We can take advantage of the fact that all edge-weights are in this range, to improve the asymptotic running time of MST algorithms. This is illustrated in the following problems.

   (a) In 3-4 sentences, describe the modifications you would make to the implementation of Prim's algorithm so that it runs in $O(m)$ time. Then present a brief running time analysis of your modified algorithm showing that it runs in $O(m)$ time.

   (b) In 3-4 sentences, describe the modifications you would make to the implementation of Kruskal's algorithm so that it runs in $O(m + n \log n)$ time. Then present a brief running time analysis of your modified algorithm showing that it runs in $O(m + n \log n)$ time.

6. Recall that a standard (First In First Out) queue maintains a sequence of items subject to the following operations.

   - `Push`(x): Add item x to the end of the sequence.
   - `Pull`(): Remove and return the item at the beginning of the sequence.
   - `Size`(): Return the current number of items in the sequence.

   It is easy to implement a queue using a doubly-linked list, so that it uses $O(n)$ space (where n is the number of items in the queue) and the *worst-case* time for each of these operations is $O(1)$.

   Consider the following new operation, which removes every *tenth* element from the queue, starting at the beginning, in $\Theta(n)$ worst-case time.

```
Decimate():
  n ← Size()
  for i ← 0 to n − 1
    if i mod 10 = 0 then
      Pull()    (Comment:the result is discarded)
    else
      Push(Pull())
```

Prove that in any intermixed sequence of `Push`, `Pull`, and `Decimate` operations, the amortized cost of each operation is $O(1)$.
(**Hint:** Use a "charging argument" in your proof, where the total cost of each operation is "charged" in some appropriate way to the elements in the queue. The prove that each element is charged at most a constant amount. From here it is just one step to the claim you need to prove.)
