School of Computing and Information Systems
COMP90038 Algorithms and Complexity Tutorial Week 12

26–30 October 2020

Plan

Sadly this is the last tutorial session; but at least there is still the exam to look forward to. You will find a list of examinable topics on the LMS, under "exam information".

The exercises

79. Floyd's algorithm sometimes works even if we allow negative weights in a dag.

For example, for the left graph above, it will produce these successive distance matrices:

$$D^0 = D^1 = D^2 = \begin{bmatrix} 0 & 4 \\ -3 & 0 \end{bmatrix}$$

What happens for the right graph above? What do D^0, D^1 and D_2 look like? Explain why D^2 ends up giving an incorrect result in this case (but not in the previous case).

80. We are given a sequence of "connection points" spaced out evenly along a straight line. There are n white, and n black points, in some (random) order.

The points are spaced out evenly, so that the distance between two adjacent points is 1.

The points need to be connected, so that each white point is connected to exactly one black point and vice versa. However, the total length of wire used must be kept as small as possible.

Consider the following (greedy) algorithm to solve the problem:

$k \leftarrow 1$
while there are still unconnected points **do**
    create all possible connections of length k
    $k \leftarrow k + 1$

Argue the correctness of this algorithm, or, alternatively, devise an example that proves that it may not produce an optimal wiring.

81. Recall the definition of the knapsack problem. Given a set of items $S = \{i_1, i_2, \ldots i_n\}$ with

- weights: $w(i_1), w(i_2), \ldots, w(i_n)$
- values: $v(i_1), v(i_2), \ldots, v(i_n)$

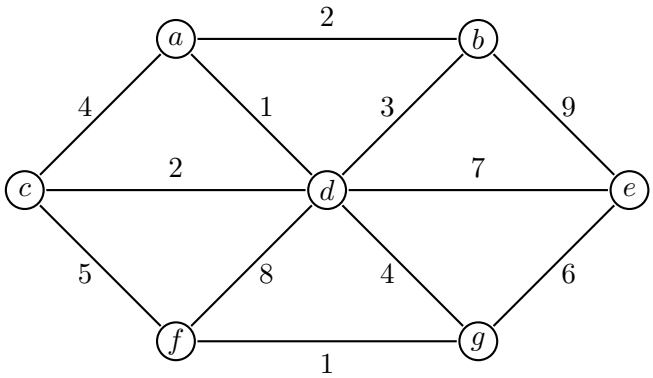
and a knapsack of capacity W, find the most valuable selection of items that will fit in the knapsack. That is, find a set $I \subseteq S$ such that $\sum_{i \in I} w(i) \leq W$ and so that $\sum_{i \in I} v(i)$ is maximised.

Define the *benefit* of an item i to be the rational number $v(i)/w(i)$. Consider the following greedy approach to the problem:

Let $A[1]...A[n]$ hold the items from S, in decreasing order of benefit
$val \leftarrow 0$
$weight \leftarrow 0$
$k \leftarrow 1$
while $k \leq n \wedge weight + w(A[k]) \leq W$ **do**
    select $A[k]$
    $val \leftarrow val + v(A[k])$
    $weight \leftarrow weight + w(A[k])$
    $k \leftarrow k + 1$

That is, at each step, from the remaining items we simply pick the one that has the greatest benefit. Give a simple example to show that this greedy algorithm does not solve the knapsack problem.

82. Work through Prim's algorithm for the graph below. Assume the algorithm starts by selecting node a. Which edges are selected for the minimum spanning tree, and in which order?



83. Use Dijkstra's algorithm to find the shortest paths for node e in the previous question's graph. That is, run the algorithm to determine the length of the shortest path from e to v, for all seven nodes v. Is the shortest path from e to b part of the graph's minimum spanning tree?

84. Lemuel Gulliver wishes to compress the string "all␣big␣endians␣and␣all␣small␣endians". Help him by building a Huffman tree for the string (there may be several valid trees) and assign a binary code accordingly, to each of the eleven characters involved (we have used ␣ to make each space character visible). The frequencies are:

a	b	d	e	g	i	l	m	n	s	␣
6	1	3	2	1	3	6	1	5	3	6

How many bits are required for the encoded string?