Advanced Algorithms

Exam Assignments

Simonas Šaltenis

9 June 2015

Full name:	
CPR-number:	
E-mail at student.aau.dk:	

This exam consists of a set of multi-choice questions and an additional exercise. There are four hours available. Mark your answers to the multi-choice questions on these pages. Remember to put your name and your CPR number on any additional sheets of paper you hand in.

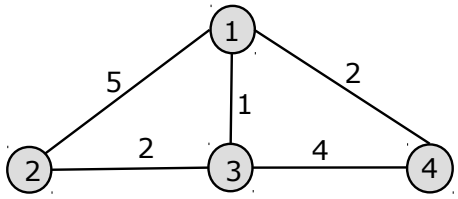
- *Read* carefully *the text of each exercise before solving it!*
- *For exercise 2, it is important that your solutions are presented in a readable form. This will make it easier to understand your answers. In particular, you should provide precise descriptions of your algorithms. Pseudo-code is strongly preferred. It is also worth to write two or three lines describing informally what the algorithm is supposed to do, as well as to justify your complexity analyses.*

If not mentioned otherwise, the algorithms referred to in Exercise 1 are assumed to be implemented as described in the main textbook of the course.

During the exam you are allowed to consult books and notes.

Exercise 1 [50 points]

1. (*7 points*) Consider the following undirected weighted graph.

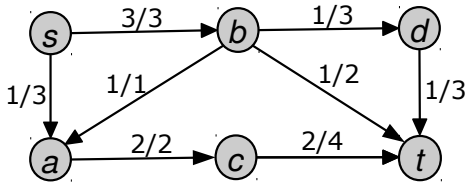


How does the **4-th** row of the distance matrix look *after* the first iteration ($k = 1$) of the Floyd-Warshall algorithm?

☐ **a)** $2, 7, 4, 0$ ☐ **b)** $2, 6, 3, 0$

☐ **c)** $2, 7, 3, 0$ ☐ **d)** $2, \infty, 4, 0$

2. (*7 points*) Consider the following flow network with some flow from s to t (flow/capacity shown on edges):

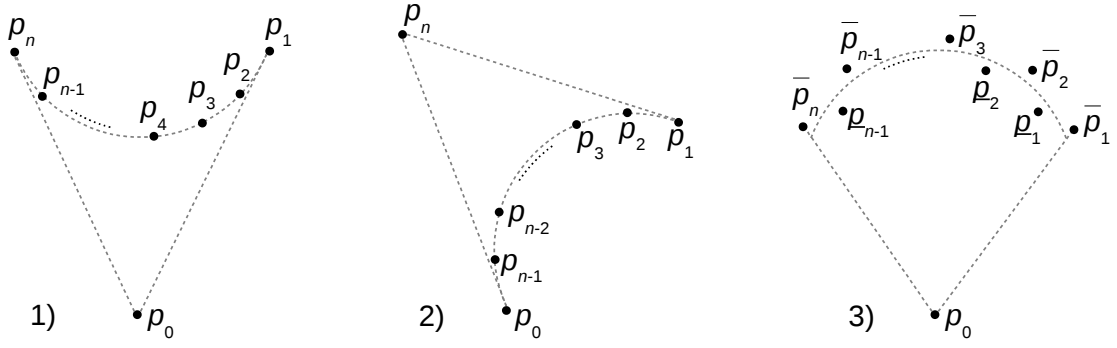


What would be the *augmenting path* chosen by the Edmonds-Karp algorithm?

☐ **a)** s, a, b, t ☐ **b)** s, b, t

☐ **c)** s, a, b, d, t ☐ **d)** there is no augmenting path

3. (*6 points*) Consider the following three sets of points (the dotted lines just illustrate the relative positions of points).



Consider running *Graham's scan* on each of the sets of points. The algorithm builds the convex hull in a stack. What will be the maximum size of this stack during the execution of the algorithm?

3.1. In Figure 1):

☐ **a)** 3 ☐ **b)** $n-1$ ☐ **c)** n ☐ **d)** $n+1$

3.2. In Figure 2):

☐ **a)** 3 ☐ **b)** $n-1$ ☐ **c)** n ☐ **d)** $n+1$

3.3. In Figure 3):

☐ **a)** $n+1$ ☐ **b)** $2n-2$ ☐ **c)** $2n-1$ ☐ **d)** $2n$

4. (*7 points*) Consider a two-dimensional *range tree* storing 16 points.

4.1. What is the total number of all the leaves in *the associated structures* of all the nodes on the second level below the root of the main tree?

☐ **a)** 8 ☐ **b)** 16 ☐ **c)** 32 ☐ **d)** 64

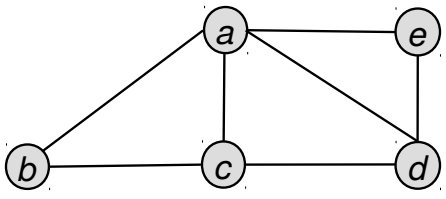
4.2. Consider one path from the root to a leaf of the main tree. What is the total number of all the leaves in *the associated structures* of the internal nodes on this path?

☐ **a)** 64 ☐ **b)** 32 ☐ **c)** 16 ☐ **d)** 30

4.3. Answer **4.2.** if the range tree stores n points.

☐ **a)** $\Theta(n)$ ☐ **b)** $\Theta(n^2)$ ☐ **c)** $\Theta(\sqrt{n})$ ☐ **d)** $\Theta(n \lg n)$

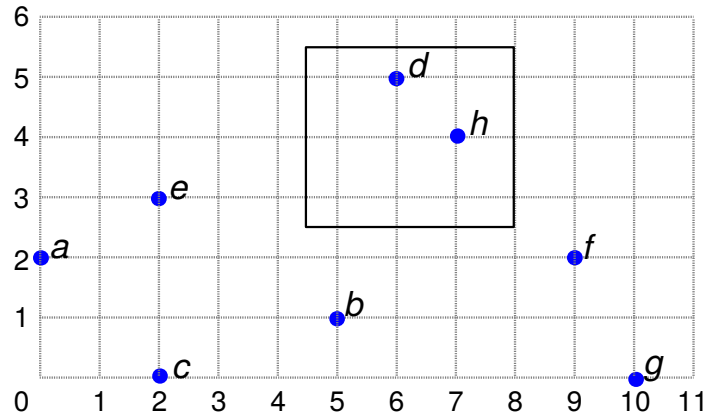
5. (*7 points*) Which of the four sets of vertices could be the result of running Approx-Vertex-Cover on this graph?



☐ **a)** a, d, b ☐ **b)** a, d

☐ **c)** b, c, d, e ☐ **d)** a, c, e

6. (*8 points*) Consider the following set of points on the plane stored in a *kd-tree*. Assume that for each internal node, the left subtree contains points with coordinates (x or y) *smaller or equal* than the split value and the right subtree points with coordinates strictly larger than the split value. Also assume that the root of the tree splits on the x coordinate (horizontal).



Which leaf nodes are accessed to process the range query shown in the figure?

☐ **a)** d, h, f ☐ **b)** d, h

☐ **c)** d, h, e ☐ **d)** d, h, b, e, f

7. (*8 points*) Consider a file of 10000 disk pages that has to be sorted. How many I/O operations (reads or writes of disk pages) are performed by the *multiway merge-sort* algorithm if 50 pages of main memory is available?

☐ **a)** 20000 ☐ **b)** 40000 ☐ **c)** 60000 ☐ **d)** 70000

Exercise 2 [50 points]

Consider an array of numbers $A[1\,..\,n]$. We want to find a *longest increasing subsequence* in this array. Formally, we are looking for a longest sequence $A[i_1] < A[i_2] < \cdots < A[i_k]$, where $1 \leq i_1 < i_2 < \cdots < i_k \leq n$.

For example, if $A = [7, 4, 5, 2, 8, 3]$ then the longest increasing subsequence is $\langle A[2], A[3], A[5]\rangle = \langle 4, 5, 8\rangle$.

Note that solution to part 1 is not necessary to solve parts 2–4.

1. (*15 points*) Assume that $A[n + 1] = \infty$, which is larger than all the other elements in the array. Write a *multithreaded* algorithm that, given an index $i\,(1 \leq i \leq n)$, finds the next element in A after i that is larger than $A[i]$.

In other words, it should return $\min\{j \mid j > i \wedge A[j] > A[i]\}$. Analyze the work, span, and parallelism of your algorithm. Note that your algorithm may on average do more work than a simple and efficient sequential algorithm, but not more (asymptotically) in the worst case.

Assume a computer with unlimited physical threads. Which algorithm do you expect to run faster—your multithreaded algorithm or an efficient sequential one—if $A[1..n]$ is sorted in the ascending order? What if $A[1..n]$ is sorted in the descending order (i.e., the algorithm returns $n+1$, independently of i)?

2. (*8 points*) Consider the following greedy algorithm to find *the length* of a longest increasing subsequence of A.

```
GREADYLIS(A[1..n])
 1  L = 0
 2  for i = 1 to n
 3       l = 1
 4       prev = i
         // Greedily construct an increasing subsequence starting at i
 5       for j = i + 1 to n
 6            if A[j] > A[prev]
 7                 l = l + 1
 8                 prev = j
 9       L = max(L, l)
10  return L
```

What is the worst-case running time of this algorithm?

Construct an example array such that this algorithm *does not* return the correct result. Explain what the correct result in your example should be and what the algorithm returns.

3. (*20 points*) Write a dynamic-programming algorithm that finds *the length* of a longest increasing subsequence of A. Analyze the worst-case running time and space of your algorithm.

4. (*7 points*) Augment your algorithm so that, in addition to the length, it also prints out a longest increasing subsequence. Do your modifications change the asymptotic worst-case running time or space requirement of your algorithm?