CONCORDIA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

COMP 6651/3: Algorithm Design Techniques

Fall 2018

Final Exam - Close book exam - 3 hours

Instructor: Professor B. Jaumard

First Name	**Last Name**	**ID#**

- If you happen to use an algorithm we saw during one of the lectures, you need to cite it, but also to describe it in detail
- Analyze your algorithm means: provide a detailed complexity analysis of your algorithm
- Design an algorithm that ... You are required to put comments for your algorithm + justifications that the algorithm is exact/heuristic
- Any answer provided without any justifications will not be considered

Question 1. Boyer-Moore substring search (20 points.)

Suppose that you run the Boyer-Moore algorithm to search for the pattern

$$A\,B\,A\,B\,A\,C$$

in the text

$$A\,B\,A\,B\,A\,B\,C\,A\,B\,A\,B\,A\,B\,C\,A\,B\,A\,B\,A\,B\,C\,A\,B\,A\,B\,A\,C\,B.$$

6 points **(a)** Give the values of the failure function, assuming a numbering from 1 to 6.

Failure Function

1	2	3	4	5	6
A	B	A	B	A	C
0	0	1	2	3	0

6 points **(b)** Give the trace of the algorithm in the grid below, circling the characters in the pattern that get compared with the text.

A	B	A	B	A	B	C	A	B	A	B	A	B	C	A	B	A	B	A	B	C	A	B	A	B	A	C	B
A	B	A	B	A	(C)																						
		A	B	A	B	(A)																					
				A	B	(A)																					
						(A)																					
							A	B	A	B	A	(C)															
									A	B	A	B	(A)	C													
											A	B	(A)	B	A	C											
													(A)	B	A	B	A										
														A	B	A	B	A	(C)								
																A	B	A	B	(A)	C						
																		A	B	(A)	B	A	C				
																				(A)	B	A	B	A	C		
																					A	B	A	B	A	C	

2 points **(c)** During the class, we discussed a naive algorithm. Provide a pattern example in which the KMP algorithm does not do any better than the naive algorithm.

> You cannot just provide a pattern: it depends on the pattern vs. the text.
>
> txt[] = "AAAAAAAAAAAAAAAAAA"
> pat[] = "XYZT"

2 points **(d)** Provide a pattern example, which is a best possible case for the KMP algorithm.

> Again, you cannot just provide a pattern: it depends on the pattern vs. the text.
>
> The naïve pattern searching algorithm does not work well in cases where we see many matching characters followed by a mismatching character. Following are some examples.
>
> txt[] = "AAAAAAAAAAAAAAAAAB"
> pat[] = "AAAAB"
>
> txt[] = "ABABABCABABABCABABABC"
> pat[] = "ABABAC" (not a worst case, but a bad case for Naive)

2 points **(e)** Assume you have to search for a particular word in a book. What is your estimate of how faster will be the KMP algorithm in comparison to the naive algorithm?

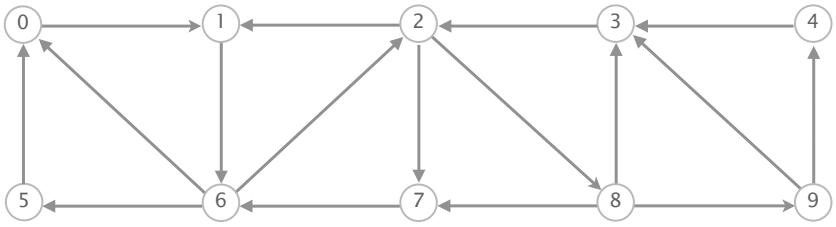
> KMP will be faster. How much? It depends on the length of the word, and on the repetition of sub-patterns inside the word.

2 points **(f)** Can you give an example of an application where the KMP algorithm performs significantly better than the naive algorithm?

> Knuth-Morris-Pratt: pre-analyzes the pattern and tries to re-use whatever was already matched in the initial part of the pattern to avoid having to rematch that. It works quite well if the alphabet is small, e.g., DNA bases, as you get a higher chance that your search patterns contain reuseable subpatterns.

Question 2. Elementary Data Structures (20 points.)

Consider the following digraph. Assume the adjacency lists are in sorted order: for example, when iterating through the edges pointing from 2, consider the edge $2 \to 1$ then $2 \to 7$, and $2 \to 8$ next.



2 points **(a)** Run depth first search on the digraph, starting with node 0. List the order in which you first visit the nodes.

_0_	_1_	_6_	_2_	_7_	_8_	_3_	_9_	_4_	_5_

2 points **(b)** List the vertices in preorder

_0_	_1_	_6_	_2_	_7_	_8_	_3_	_9_	_4_	_5_

2 points **(c)** Which order defines a topological order?

reverse post order

2 points **(d)** List the vertices in the order of (c)

Note that this is not a "valid" topological order, as there are circuits (loops).

postorder	_7_	_3_	_4_	_9_	_8_	_2_	_5_	_6_	_1_	_0_
reverse postorder	_0_	_1_	_6_	_5_	_2_	_8_	_9_	_4_	_3_	_7_

A 3-heap is an array representation of a complete ternary tree, where the key in each node is greater than (or equal to) the keys in each of its children.

(e) Perform a delete-the-maximum operation on the following 3-heap, which is the level-order traversal of a complete ternary tree, using 1-based indexing.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
--	88	33	77	66	10	30	25	23	60	75	14	21	50	9	7

3 points Fill in the table below to show the resulting 3-heap, circling any entries that change.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	(77)	33	(75)	66	10	30	25	23	60	(7)	14	21	50	9	--

(f) Given the array index k of a key, what are the indices of its three (potential) children as a function of k?

1 point

$3k+1, 3k+2, 3k+2$

(g) What is the maximum number of compares for a delete-the-maximum operation as a function of the number of keys N in the data structure? Circle the best answer.

2 points

At each node, we need to find the maximum of 4 elements, hence 3 comparisons, in a tree of height $\log_3 N$.

~ 1 $\sim \log_2 N$ $\sim \log_3 N$ $\sim 2\log_3 N$ $\sim 2\log_2 N$ $\underline{\sim 3\log_3 N}$ $\sim N$

6 points **(h)** Provide the algorithm (pseudo-code) for building a heap in $O(n)$

```
MAX_HEAPIFY(A, i)
1. ℓ ← LEFT(i)
2. r ← RIGHT(i)
3. if ℓ ≤ heap_size[A] and A[ℓ] > A[i]
4.       then largest ← ℓ
5.       else largest ← i
6. if r ≤ heap_size[A] and a[r] > A[largest]
7.       then largest ← r
8. if largest ≠ i
9.       then exchange A[i] ↔ A[largest]
10.           MAX_HEAPIFY (A, largest)

BUILD_MAX_HEAP(A)
1. heap_size[A] ← LENGTH[A]
1. for i ← ⌊LENGTH[A]/2⌋ downto 1
3. do MAX_HEAPIFY(A, i)
```

Question 3. (20 points.)

There are M job applicants and N jobs. Each applicant has a subset of jobs that he/she is interested in. Each job opening can only accept one applicant and a job applicant can be appointed for only one job.

(a) Find an assignment of jobs to applicants in such that as many applicants as possible get jobs. Show that it can be reduced to finding a maximum matching in a bipartite graph. Write down a concise description of your algorithm. (A detailed self-content description of the algorithm is required).

8 points

Graph def.: 2 points

Build a bipartite graph $G = (V = A \cup J, E)$ such that the first set of nodes (A) is associated with the applicants, and the second set of nodes (J) is associated with the jobs. There is an edge $e \in E$ between a job and an applicant if the job belongs to the list of jobs of interest for the applicant. Note that $|E| = O(|A| \times |J|)$.

Then finding an assignment of jobs to applicants such that as many applicants as possible get jobs is equivalent to finding a matching in G.

An alternating path with respect to M alternates between edges in M and in $E \setminus M$.

An augmenting path with respect to M is an alternating path with first and last vertices exposed.

A vertex is exposed if it is not the endpoint of any edge in the matching.

Algo 1: 3 points

Algorithm Matching

1. Start with any matching M (lets say $M = \{\}$)
2. As long as there exists an augmenting path with respect to M:
3. Find augmenting path P with respect to M
4. Augment M along P: $M' \leftarrow M \Delta P$
5. Replace M with the new M'

Algo 2: 3 points

Algorithm: Finding an Augmenting Path in a bipartite graph

1. Direct all edges in the matching from J to A, and all edges not in the matching from A to J
2. Create a node s that connects to all exposed vertices in set A
3. Do a Breadth First Search to find an exposed vertex in set J from node s.

2 points

(b) Analyze its complexity, provide the best possible $O()$ complexity.

Complexity: 1 point

Justification: 1 point

Complexity: $O(\min\{|V|, |A| + |J|\} \times (|A| \times |J|))$.

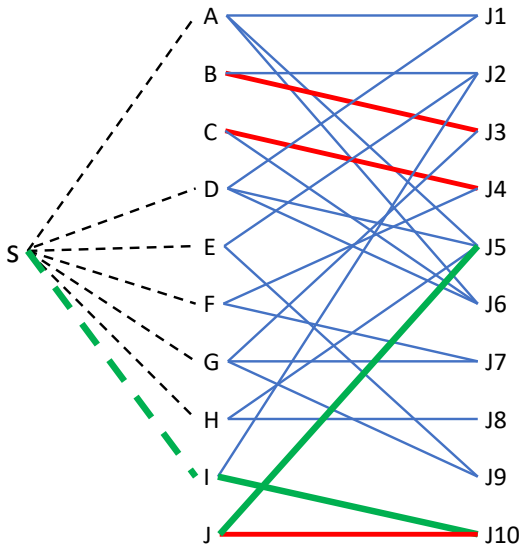
Justification: Each augmenting path costs a BFS. One BFS has a complexity of $O(|E|)$. How many iterations? No more than the size of the maximum matching ($M^\star$), i.e., at most $\min\{|V|, |A|\}$.
Hence, overall complexity: $\min\{|V|, |A|\} \times |A| + |J|\} = mn \times \underbrace{\min\{m, n\}}_{\geq \text{ size of } M^\star}$.

Best known complexity: Chandran and Hochbaum (2011) runs in time that depends on the size (k) of the maximum matching. Assuming $|X| < |Y|$, we have:

$$O\left(\min\{|X|k, E\} + \sqrt{k}\min\{k^2, E\}\right).$$

Chandran, Bala G.; Hochbaum, Dorit S. (2011), Practical and theoretical improvements for bipartite matching using the pseudoflow algorithm.
However, the theoretically efficient algorithms listed above tend to perform poorly in practice.

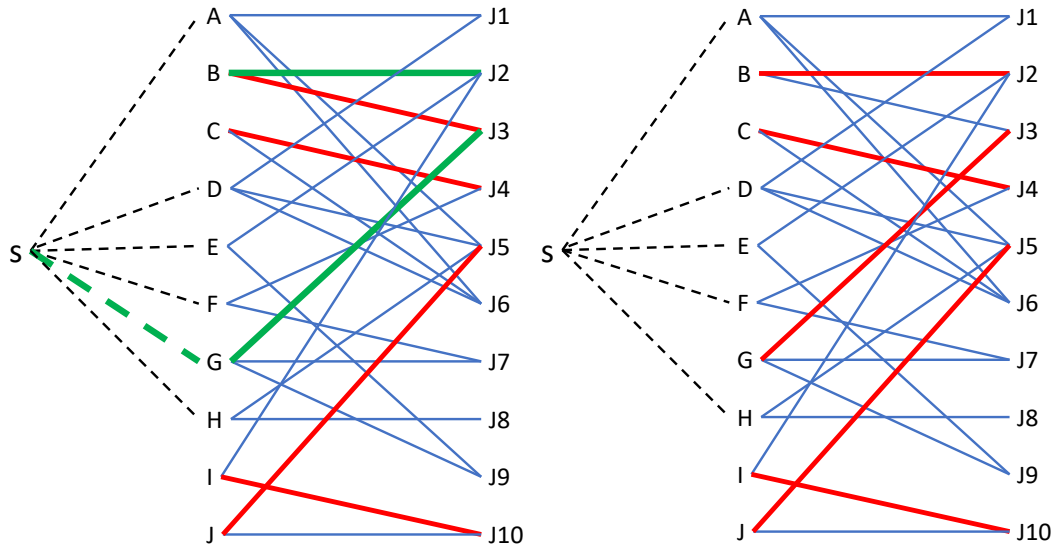
(c) Apply the algorithm to the following example starting with an empty matching, i.e., $M = \{(C, J4), (J, J10)(B, J3)\}$.
Go through the first 2 iterations
A = Set of applicants = $\{A, B, C, D, E, F, G, H, I, J\}$.
J = Set of jobs = $\{J1, J2, J3, J4, J5, J6, J7, J8, J9, J10\}$.
Interests =
$A \rightsquigarrow J1, J5, J6$ $B \rightsquigarrow J2, J3$ $C \rightsquigarrow J4, J6$
$D \rightsquigarrow J1, J5, J6$ $E \rightsquigarrow J2, J9$ $F \rightsquigarrow J7, J4$
5 points $G \rightsquigarrow J3, J7, J9$ $H \rightsquigarrow J8, J5$ $I \rightsquigarrow J2, J10$ $J \rightsquigarrow J5, J10$.



Iteration 1

$M_0 = \{(C, J4), (J, J10)(B, J3)\}$
$M_1 = \{(C, J4), (I, J10), (J, J5)(B, J3)\}$

Iteration 2



$M_2 = \{(C, J4), (I, J10), (J, J5), (G, J3), (B, J2)\}$

(d) We now add weights on the edges. Find an assignment of jobs to applicants in such that as many applicants as possible get jobs, and such that the sum of the weights is maximum (assuming weights express the preference of the applicants to the jobs.) Show that the resulting problem can be solved after being reformulated as a maximum flow problem. Give a concise definition of the associated network graph (nodes/links) and the link capacities.

5 points

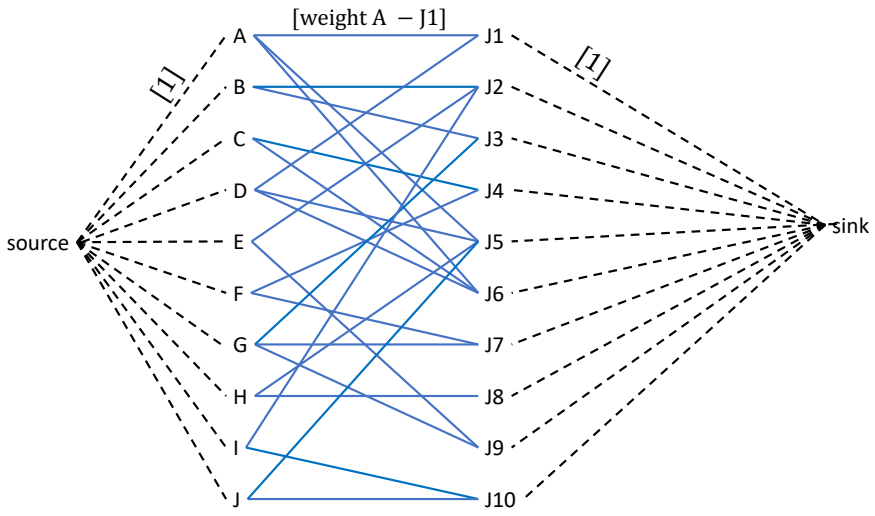
Integer: 1 point

Capacity: 2 points

Topology: 2 points

If weights are not integer values, multiply them by 10^k, with k as small as possible, so that weights become integer. This guarantees that optimal flow has integer values.

Capacity of the links from source to nodes of A, and of links from nodes of J to sink: 1, as otherwise we do not guarantee that we will get a matching.



Question 4. (20 points.)

Given two arrays A and B with n character each. The task is to make these two arrays identical, i.e., for each $1 \leq i \leq N$, we want to make $A_i = B_i$. In a single operation, you can choose two characters x and y, and replace all the occurrences of x in both the arrays with y. Notice that regardless of the number of occurrences replaced, it will still be counted as a single operation.

10 points

(a) Design an algorithm that outputs the minimum number of operation required. Write an algorithm that highlights the union and find operations.

MakeIdenticalArrays(A, B)

$S \leftarrow$ disjoint-set containing a set for each character

ANSWER $= 0$ (at the end, will contain the number of operations)

For $i = 1$ to n **do**
    If FIND(A_i) $\neq$ FIND(B_i) **then**
        ANSWER $\rightarrow$ ANSWER $+ 1$
        UNION (FIND(A_i), FIND(B_i))
    EndIf
EndFor

Return ANSWER

10 points

(b) Analyze its time and space complexity.

1 point

Space complexity: $O(n)$.

2 points

Justification space complexity:
We use limited an extra space of length n for the bucket sort, plus the counter.

2 points

Time complexity: $O(n \log n)$.

5 points

Justification time complexity:

- Computation of S
  Can be done in $O(n)$. One pass to identify the repetitions of characters and move all the same characters in the set (use, e.g., bucket sort). Second pass to assign ID to the sets (or buckets).
- For the **For** loop
  Assume list based data structure.
  $2 \times n$ FIND operations: $O(1)$ for each find operation $\rightsquigarrow O(n)$
  $\leq n$ UNION operations: always move elements from the smaller set to the larger set.
  - Each time an element is moved it goes to a set of size at least double its old set.
  - Thus, an element can be moved at most $O(\log n)$ times.

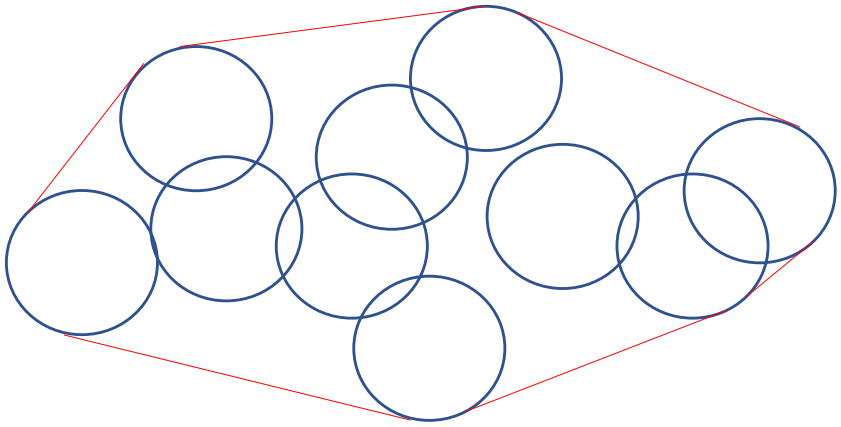
$$1 + 2 + 2^2 + 2^3 + \cdots + 2^k = 1 + 2(2^k - 1) \leq n$$

$$\hookrightarrow \qquad k \leq \log_2(\frac{n-1}{2} + 1) = O(\log_2 n)$$

Question 5.

(20 points.)

In many situations we need to compute convex hulls of objects other than points. Let S be a set of (possibly intersecting) unit circles in the plane. We want to compute the convex hull of S.



(a) Show that the boundary of the convex hull of S consists of straight line segments and pieces of circles in S.

5 points

Using the definition of a convex hull, the convex hull of S will be the intersection of all closed half-planes containing all of S.

Proof is by induction on the number of circles.

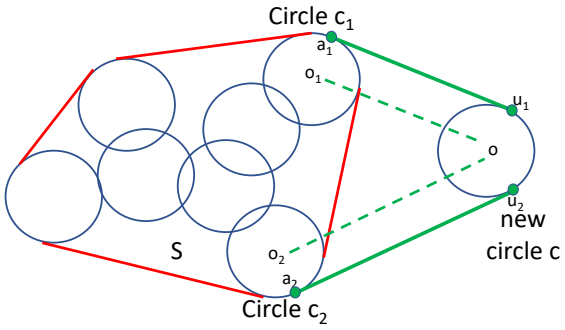
If $n = |S| = 1$, there is a single circle which is the convex hull itself (and smaller form will not contain the entire sole circle), which satisfies the boundary condition.

Assume any convex hull boundary for $n \geq 2$ is formed of straight line segments and pieces of circles in S, and denote that convex hull $\text{CH}(S)$. Now we add a new unit circle C'. If $c \subset S$, then the new convex hull for the $n+1$ circles is simply $\text{CH}(S)$, which satisfies the boundary condition by the induction assumption. Otherwise, we will construct the new convex hull for the whole $n+1$ unit circles as follows:

Pick some point p such that $p \notin c, p \notin \text{CH}(S)$ and $[p, o] \cap \text{CH}(S) = \emptyset$, where o is the center of circle C'. Now find the points o_1 and o_2 out of all circle center points of the circles in S with the smallest and largest angles $\angle poo_1$ and $\angle poo_2$. Let the two circles with centers O_1 and o_2 be c_1 and c_2, respectively. Find the line segments $[a_1, u_1]$ and $[a_2, u_2]$, such that a_1 is on the perimeter of c_1, u_1 is on the perimeter of C' and the segment $[a_1, u_1]$ is parallel to $[o_1, o]$, and $[a_2, u_2]$ in a similar fashion is defined. The closed shape we got, that is defined by the perimeter:

$$u_1 \xrightarrow{\text{over the perimeter of } c} u_2 \to a_2 \xrightarrow{\text{over the perimeter of } c} a_1 \to u_1$$

is the new convex hull for all $n+1$ unit circles, where $u_1 \to u_2$ is a part of the circle C', u_2 and a_2 are straight line segments and $a_2 \to a_1$ is on the original perimeter of the convex hull of the n circles, thus by assumption is constructed of only straight line segments and parts of circles. Therefore the new convex hull is also constructed as required.



o

Correctness for the convex hull construction above: denote the new set CH^+. To show it is convex, it is sufficient to show that for any pair of circles c, c' where C' is the newly added circle and c' is one of the original circles in $\text{CH}(S)$, $\forall u \in C', \forall a \in c', [u, a'] \in \text{CH}^+$. But in the way we selected $[a_1, u_1], [a_2, u_2]$, each tube from any c' to C' is fully contained in CH^+. Moreover, if these segments were note selected to be parallel:

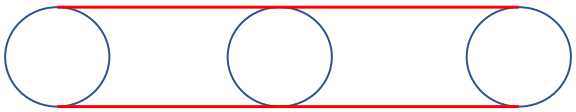
- If they were towards "inside" CH^+, we would have had a concave shape.
- If they were towards "outside" CH^+, we could have narrowed the shape in area up to the point of tangency.

Therefore CH^+ is the convex hull of the given circles.

5 points

(b) Show that each circle can occur at most once on the boundary of the convex hull.

This in fact is not correct, as in the following example, where the 2 tangent points of the middle circle appear on the boundary of the convex hull:



o

But assuming the meaning is that no circle can appear twice on the boundary of the convex hull, where each appearance includes more than one point, following is a proof:

If S includes only one unit circle C' then the convex hull $\text{CH}(S) = c$ itself, therefore appears only once on the boundary of $\text{CH}(S)$ (the entire perimeter of C'). If it includes more than one circle, assume that it appears twice on the boundary of $\text{CH}(S)$ such that each appearance is more than one point. Let the points p, q, r, s be the points on the circle C' such that from these points the boundary of $\text{CH}(S)$ continues towards away from C'. Assume that the points p, q are on a sector of C' that is inside $\text{CH}(S)$ (i.e., the sector p, q on C' is not on the boundary of $\text{CH}(S)$), then lines go out from p, q to continue the boundary of $\text{CH}(S)$. At the best case, there is only one additional circle at that direction, such that the lines that go out from p, q away from C' to that circle are tangent to that circle. But by the definition of p, q, the sector (p, q) is shorter than the sector between the tangent points on the other circle, and so it concludes that $\text{CH}(S)$ will not be convex, a contradiction: Thus any circle can appear only once on the boundary of $\text{CH}(S)$.

(c) Let S^{IN} be the set of points that are the centers of the circles in S. Show that a circle C in S appears on the boundary of the convex hull if and only if the center o of the C circle lies on the convex hull $\text{CH}^{\text{IN}}(S)$ of S^{IN}.

5 points

Let $\text{CH}(S)$ be the convex hull of S and $\text{CH}^{\text{IN}}(S)$ the convex hull of S^{IN}.

First assume the circle $C \in S$ appears on the boundary of $\text{CH}(S)$, and let o be its center. Assume that o is not on the boundary of $\text{CH}^{\text{IN}}(S)$, and look at an arbitrary pair of vertices of $\text{CH}^{\text{IN}}(S)$, say o_1, o_2. By definition, o is contained in the intersection of the convex sets defined by all lines o_1o_2. Then, for every o_1, o_2, we can pump to the unit circles, then they all cover the area around them distant by 1. But since we pumped o_1, o_2 and o by the same measurement, then the new boundary is a line tangent to the circles formed around o_1, o_2 and the circle around o, C, does not intersect that line, hence it is still contained in the convex set generated by the tangent above. Therefore C is not on the boundary of $\text{CH}(S)$, in contradiction to the assumption, so must be on the boundary of $\text{CH}^{\text{IN}}(S)$.

Now assume o is a vertex of $\text{CH}^{\text{IN}}(S)$, and assume C is not on the boundary of $\text{CH}(S)$. In a similar fashion to the previous part, this means C is contained by the intersection of all convex sets defined by the lines tangent to pairs of circles on the boundary of $\text{CH}(S)$. Then we can pump down the circles to their center, and o will be fully contained by pumped-down shape (which is now a polygon), therefore will not be on its boundary, in contradiction with the assumption. Therefore C must be on the boundary of $\text{CH}(S)$.

5 points (d) Give an $O(n \log n)$ algorithm for computing the convex hull of S.

Given what is previously proven, to find the convex hull of a set of unit circles S, denoted by $\text{CH}(S)$, we do as follows:

- If S contains 1 circle, return it; if it contains 2 circles, return the tube formed by crossing 2 tangent lines between them.
- Otherwise, find the convex hull of the set center points of the circles in , using Grahams scan, denoted by $\text{CH}^{\text{IN}}(S)$.
- For any 3 consecutive vertices of $\text{CH}^{\text{IN}}(S)$, denoted by p, q and r, add to the boundary of $\text{CH}(S)$:
  - The line segment $[p', q']$, which is parallel to $[p, q]$, distant by 1 from it and forms a rectangle $pp'q'q$ towards outside of $\text{CH}^{\text{IN}}(S)$.
  - The line segment $[p", r']$, defined in a similar fashion with respect to q, r.
  - The circle sector $q' \rightarrow q"$ of the unit circle with the center point q, which goes outside the shape were building (i.e. if is the intersection point of the lines $p', q', q", r'$, then the sector contained in the triangle $q'sq"$).

    Note: the boundary can be sufficiently defined by the finite set of points $p', q', q", r'$ (derived from all $\text{CH}(S)$-vertices triplets) and a mapping of each part to either a line segment (like $p' \rightarrow q', q" \rightarrow r'$) or a unit-circle sector (like $q' \rightarrow q"$).
- Return $\text{CH}(S)$.

This algorithm clearly pumps the convex set it creates for the circles center points, as required. The first part of running Grahams scan is $O(n \log n)$, and the rest is linear, concluding to $O(n \log n)$ in total, as required.