

CSE 101
Practice Midterm: Spring, 2004

Answer all questions. Give informal (at least) proofs for all answers. Grading will be on completeness and logical correctness, as well as correctness. For the data structure question, the efficiency of your solution will also be taken into account.

Analyzing loops-10pts Consider the following iterative algorithm, that uses an $\Theta(I)$ time procedure $proc(I)$, which does not change I .

Algorithm(n : positive integer);

1. begin;
2. $I \leftarrow 1$;
3. While $I \leq n$ do:
4. begin;{while}
5. FOR $J = 1$ TO $\lfloor n/I \rfloor$ do $proc(I)$
6. $I \leftarrow 2 * I$.
7. end;{while}
8. end;

Give a time analysis, up to Θ , for this algorithm.

Correctness proofs Say that a graph is d -dense if every node has at least d edges adjacent to it. The following algorithm strategy determines whether graph G has a non-empty d -dense sub-graph G' . After the algorithm, a correctness proof with some gaps is given. Fill in the gaps to form a complete proof. (The gaps are numbered with Roman numerals. On your exam answers, write the phrases that should be in the gaps, in that order.)

DenseSubgraph(G: undirected graph, d: positive integer)

1. While there is a node x of degree $< d$ do:
2. $G \leftarrow G - \{x\}$.
3. If G is empty return False, else return True.

We need to prove that, if G has a d -dense sub-graph G' , then the above algorithm returns I , and conversely, that, if II , then III .

Assume that G has a d -dense subgraph $S \subseteq G$. We prove the algorithm by IV using the following loop invariant: Let G_t be the remaining graph after t iterations of the while loop. Then $S \subseteq G_t$.

Since after 0 loops, $G_0 = V$, the VI case $t = 0$ is clear.

For the induction step, assume the hypothesis holds for t , i.e., that VII . We will show it holds for $t + 1$, i.e., that $VIII$.

Let x be the node deleted in the $t + 1$ 'st iteration. By the definition of the algorithm $deg(x) < d$. By the definition of d -dense, all nodes $y \in S$ have $deg(y) \geq d$. Thus, $x \notin S$, so $S \subseteq G_t - \{x\} = VIII$. Thus, we have proved our goal. Then, by induction, the invariant holds after any number of iterations.

In particular, if the loop halts after t iterations, XIV . Since $S \neq \emptyset$ by the definition of d -dense, $XV \neq \emptyset$. Thus, at this point, the algorithm returns XVI as we wanted to prove.

Conversely, assume the algorithm returns $XVII$. Then when the loop halts, we have $G_t \neq XVIII$. Since the loop has halted, every node x in G_t has $deg(x) < d$. Thus, G_t is a XX of G , which proves the converse direction.

Data structures and efficient versions of algorithms 10 pts: Union of intervals: We are given a list of (not necessarily disjoint) intervals of the real line, $I_1 = [s_1, f_1] \dots I_n = [s_n, f_n]$, with each $s_i < f_i$. We want to express the union of these intervals as a set of disjoint intervals. A strategy to do this is:

1. Initialize an empty set of intervals. *Disjoint*
2. Initialize S as the smallest start time of an interval, and F as the finish time of this interval.
3. While there is an interval I_j that ends after F ($F < f_j$) do:
 4. IF there is an interval I_j so that $s_j \leq F < f_j$ THEN $F \leftarrow f_j$.
 5. ELSE do:
 6. Add $[S, F]$ to *Disjoint*.
 7. Find the interval I_j with the smallest s_j so that $F < s_j$
 8. $S \leftarrow s_j, F \leftarrow f_j$.
 9. Add $[S, F]$ to *Disjoint*.
10. Return *Disjoint*

Give an efficient implementation of this strategy, specifying pre-processing and data structures used. Give a time analysis for your version.

Example: Input: $[7,9], [1,6], [2,8], [4,6], [10,12], [11,21]$.

The first interval to start is $[1, 6]$, so we start S at 1 and F at 6. Interval $[2,8]$ intersects with and extends $[S, F]$, so we unite it with $[1,6]$ to get $F = 8$. Now interval $[7,9]$ intersects with and extends our interval, so we unite it with $[1,8]$ to get $[1,9]$. Now, any other interval is either contained by or disjoint from $[1,9]$, so we put $[1,9]$ in the output. The first interval to start after 9 is $[10,12]$, so those are our new values of S, F . This intersects with $[11,21]$, so we unite it with our interval to get $[10,21]$. No intervals end after 21, so that is as far as we go, so we add $[10,21]$ to *Disjoint* to get our final output: $[1,9], [10,21]$.

Divide-and-Conquer Recurrence: 10 points Consider the following recursive algorithm. Its input is an array of (not necessarily positive) integers. $A[1..n]$ The goal is to find the maximum possible sum of a consecutive sub-sequence $A[i..j]$ of elements of the array, $\max_{1 \leq i \leq j \leq n} \sum_{k=i}^j A[k]$.

MaxConsSum[A[1..n]]

1. IF $n = 0$ return 0.
2. IF $n = 1$ return $\max(0, A[1])$.
3. $S_1 \leftarrow \text{MaxConsSum}[A[1..n/2]]$
4. $S_2 \leftarrow \text{MaxConsSum}[A[n/2 + 1..n]]$
5. $\text{SumLeftFromCenter} \leftarrow 0$
6. $\text{BestSLFC} \leftarrow 0$
7. FOR $I = n/2$ down to 1 do:
 8. $\text{SumLeftFromCenter} \leftarrow \text{SumLeftFromCenter} + A[I]$
 9. IF $\text{BestSLFC} < \text{SumLeftFromCenter}$ THEN $\text{BestSLFC} \leftarrow \text{SumLeftFromCenter}$.
10. $\text{SumRightFromCenter} \leftarrow 0$
11. $\text{BestSRFC} \leftarrow 0$
12. FOR $I = n/2 + 1$ TO n do:
 13. $\text{SumRightFromCenter} \leftarrow \text{SumRightFromCenter} + A[I]$
 14. IF $\text{BestSRFC} < \text{SumRightFromCenter}$ THEN $\text{BestSLFC} \leftarrow \text{SumRightFromCenter}$.
15. $S_3 \leftarrow \text{BestSLFC} + \text{BestSRFC}$
16. Return $\max(S_1, S_2, S_3)$.

Give a recurrence for the time $T(n)$ taken by the above algorithm Use the recurrence to give a time analysis up to order.