CSE 101

Midterm Examination

W.A. Burkhard & E. Ettinger

May 22, 2008

One handwritten study sheet is allowed.

name Solution

departmental survey completed ☐

student identifier ______________________

1. ______________

2. ______________

3. ______________

4. ______________

total ______________

Run time calculation (*10 points*) name ______________________

Indicate the big-O run time for each of the following; justify your answer.

1.a $T(n) = 3T(n/2) + n$

$a=3 \quad b=2 \quad d=1$

$\log_2 3 > 1 \implies \boxed{T(n) = O(n^{\log_2 3})}$

1.b
```
function showit ( n )
  if ( n > 0 )
    print_line ( "one line" )
    showit ( n/3 )
    showit ( n/3 )
```

$T(n) = 2T(\frac{n}{3}) + O(1)$

$a=2 \quad b=3 \quad d=0 \implies \boxed{T(n) = O(n^{\log_3 2})}$

$\log_3 2 > 0$

1.c $T(n) = 2T(n-1) + 1$

$$T(n) = 2T(n-1) + 1$$
$$= 2[2T(n-2) + 1] + 1$$
$$\vdots$$
$$= 2^n T(0) + \sum_{i=0}^{n-1} 2^i = \boxed{O(2^n)}$$

1.d $T(n) = T(\sqrt{n}) + 1$ hint: think of n as a power of 2.

Let [redacted] $2^m = n$

$T(2^m) = T(2^{m/2}) + 1$

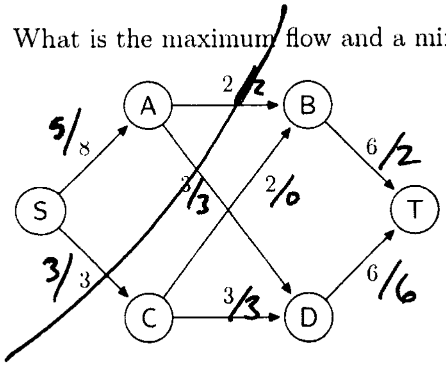
Let $R(m) = T(2^m)$

$R(m) = R(m/2) + 1 \xRightarrow{\text{by Master}} O(\log m)$

$\implies \boxed{T(n) = O(\log \log n)}$

Miscellaneous (*10 points*) name ____________________

2.a What is the maximum flow and a minimum cut in this network flow graph.



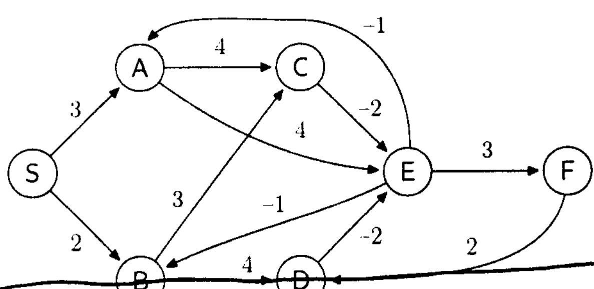
Max-flow is 8

Min-cut is

$S = \{s, A\}$ and $T = \{B, C, D, T\}$

~~2.b Dijkstra's algorithm utilizes a priority queue, with the additional method *decreasePrio*(item , newPrio), to locate the next item to process. We used an extra array, indexed by item, to locate the associated node within the the priority queue tree structure. Why is the decreasePrio method an O(log|V|) operation rather than O(1)?~~

2.c What are the shortest distances from node S in this graph?



	S	A	B	C	D	E	F
$\delta(v)$	0	2	2	5	6	3	6

dist(s,k):

node \ k	0	1	2	3	4	5	6
S	0	∞	∞	∞	∞	∞	∞
A	∞	3	∞	6	2	9	5
B	∞	2	∞	6	2	9	5
C	∞	∞	5	∞	9	5	12
D	∞	∞	6	∞	10	6	13
E	∞	∞	7	3	10	6	4
F	∞	∞	∞	10	6	13	9

Algorithm design (*10 points*) name ______________________

3. **Divide and Conquer.** You are given an list of distinct number $a_1, a_2, \ldots, a_n$ and you are to determine how "unsorted" the array is. We say the pair (a_i, a_j) is *inverted* if $i < j$ and $a_i > a_j$. You are to count how many pairs (a_i, a_j) are inverted. For example, the list 2,4,1,3,5,6 has three inversions (2,1), (4,1), (4,3). Give a divide and conquer algorithm that is better than the naïve $O(n^2)$ algorithm which would compare every pair of numbers.

Algorithm

- Run MergeSort & count swaps.

MergeSort (a[1...n]):
  if n>1
    return merge(mergesort(a[1...⌊n/2⌋]), mergesort(a[⌊n/2⌋+1...n]))
  else
    return a

Merge (x[1...k], y[1...ℓ])
  if k=0 return y[1...ℓ]
  if ℓ=0 return x[1...k]
  if x[1] ≤ y[1] return x[1] ∘ merge(x[2...k], y[1...ℓ])
  else: count++ and return y[1] ∘ merge(x[1...k], y[2...ℓ])

Global var init. to 0.

Correctness justification

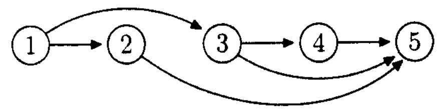
Count will contain the number of inversions in the entire list. It counts how many swaps from the LHS to the RHS occur in MergeSort which is exactly what is desired.

Runtime analysis

Same as MergeSort: $T(n) = 2T(n/2) + O(n)$
$= \boxed{O(n \log n)}$

Algorithm design (*10 points*) name ____________________

4. **Dynamic Programming.** You are given a connected DAG containing n vertices that is topologically sorted (i.e. edges are of the form (v_i, v_j) where $i < j$), moreover every v_i, except v_n, has at least one outgoing edge of this form. You are to determine, via dynamic programming, the length of the longest (most edges) and shortest paths (least edges) starting at v_1 and going to v_n. Here is a small prototypical DAG.



Recurrence(s) with base cases

Let $L(i)$ be the # of edges in longest path from v_1 to v_i
and $S(i)$ " shortest " "

Base: $L(1) = S(1) = 0$

Recurrences:

$$L(j) = \max\{L(i)+1 : (i,j) \in E\}$$
$$S(j) = \min\{S(i)+1 : (i,j) \in E\}$$

Correctness justification

The shortest and longest paths to v_1 are both 0. Any shortest path must be the SP to some previous node plus one edge to the destination. This is exactly the recurrence. The same holds for longest paths.

Runtime analysis

$O(|V|+|E|)$ since we examine each node and its edges exactly once.

Scratch paper do not remove from the test booklet name ______________________