CS21003 Algorithms I, Autumn 2011–12

Class test 1

Maximum marks: 20 | Date: 13-Sep-2011 | Duration: 1 hour

Roll no: ____________________ **Name:** ____

[*Write your answers in the question paper itself. Be brief and precise. Answer all questions.*]

1. Write a C function that, given an array A of size n, decides whether A is the contiguous representation of a max-heap. **(4)**

```
int isMaxHeap ( int A[], int n )
{
   int i, p;

   for (i=n-1; i>=1; --i) {
      p = (i - 1) / 2;
      if (A[p] < A[i]) return 0;
   }
   return 1;
}
```

2. You are given a rooted tree T represented by a binary tree B in the first-child-next-sibling representation. Write a C function to determine the height of the original tree T (not of the binary tree B). **(4)**

```
int height ( bintreenode *B )
{
   int h = 0, t;

   if (B == NULL) return -1;
   B = B -> left;
   while (B != NULL) {
      t = height(B);
      if (t > h) h = t;
      B = B -> right;
   }
   return h;
}
```

3. You are given the adjacency matrix A of an undirected graph G with n vertices numbered $0, 1, 2, \ldots, n-1$. Consider the following dynamic-programming algorithm to decide whether G is connected or not.

(a) Define $A^{(l)}$ to be the $n \times n$ Boolean matrix such that $A^{(l)}[i][j] = 1$ if and only if there is a path of length $\leqslant l$ from vertex i to vertex j. How can you compute $A^{(1)}$ from A? **(2)**

Solution Copy A to $A^{(1)}$, and set $A^{(1)}[i][i] = 1$ for all $i = 0, 1, 2, \ldots, n-1$.

(b) You are given $A^{(l_1)}$ and $A^{(l_2)}$. Describe how you can compute $A^{(l_1+l_2)}$. **(2)**

Solution $A^{(l_1+l_2)}[i][j] = 1$ if and only if $A^{(l_1)}[i][k] = 1 = A^{(l_2)}[k][j]$ for some $k \in \{0, 1, 2, \ldots, n-1\}$.

(c) What is the running time of your algorithm of Part (b)? **(2)**

Solution $O(n^3)$.

(d) Compute $A^{(1)}, A^{(2)}, A^{(3)}, \ldots, A^{(n-1)}$ (in that sequence). How can you use $A^{(n-1)}$ to decide whether G is connected or not? **(2)**

Solution Check whether $A^{(n-1)}[0][j] = 1$ for all $j = 0, 1, 2, \ldots, n-1$. It is not necessary to look at the entire matrix. Only one row/column will do.

(e) What is the running time of the algorithm of Part (d)? **(2)**

Solution $O(n^4)$.

(f) Design an $O(n^3 \log n)$-time algorithm based upon the computation of $A^{(n-1)}$. Since G cannot contain paths of lengths $\geqslant n$, it follows that $A^{(e)} = A^{(n-1)}$ for all $e \geqslant n-1$. **(2)**

Solution Using Part (b), compute $A^{(2)}, A^{(4)}, A^{(8)}, A^{(16)}, \ldots, A^{(2^k)}$, where $2^{k-1} < n-1 \leqslant 2^k$. Call $e = 2^k$. Check whether $A^{(e)}[0][j] = 1$ for all $j = 0, 1, 2, \ldots, n-1$.

Remark: $A^{(1)}$ is called the *reflexive closure* of A, and $A^{(n-1)}$ the *reflexive-transitive closure* of A. We will later study an $O(n^3)$-time dynamic-programming algorithm to compute $A^{(n-1)}$. Connectedness of G can, however, be decided in only $O(n^2)$ time.