CORE EXAM: ALGORITHMS PART

Department of Computer Science
New York University
Jan 21, 2003

Question 1

PART (i) (2 points)

Given the array $A[1,...,8] =< 6,10,13,5,8,3,2,11 >$, the pivot element $A[1] = 6$, and the Partition pseudo-code

```
PARTITION(A, p, q):
    1.  let x ← A[p]    (pivot)
    2.  let i ← p
    3.  for j ← p + 1 to q
    4.    do if A[j] ≤ x,
    5.         then i ← i + 1
    6.              exchange A[i] ↔ A[j]
    7.              print A
    8.  exchange A[i] ↔ A[j]
    9.  print A
    10. return i
```

Show the arrays that are printed eachtime lines 7 and 9 are executed (for a total of 4 arrays).

PART (ii) (3 points)

Assume all array elements are distinct. Given the Pseudo-Code of the quicksort program, provide the worst-case running time, $T(n)$. First, set up the recurrence equation (2 points) and then solve it (1 point)

Hint: The worst-case happens when the array input is provided sorted or reverse sorted and the Partition program splits the array with one side having zero elements.

```
QUICKSORT(A, p, r):
    1.  if p < r
    2.      then q ← PARTITION(A, p, r)
    3.           QUICKSORT(A, p, q − 1)
    4.           QUICKSORT(A, q + 1, r)
```

Initial call: QUICKSORT$(A, 1, n)$

PART (iii) (3 points) What is the best-case running time, T(n)? First, set up the recurrence equation (2 points) and then solve it (1 point)

Hint: The best-case happens when the Partition program splits the array evenly at every step.

PART (iv) (2 points)

Suppose the input array is such that the Partition program alternates, at one step splitting the array evenly (lucky) and at the next step splitting the array with one side having zero elements (unlucky). What will be the running time of quicksort for such arrays of size n?

Solution to Question 1

PART (i)

1. $A =< 6, 5, 10, 13, 8, 3, 2, 11 >$
2. $A =< 6, 5, 3, 10, 13, 8, 2, 11 >$
3. $A =< 6, 5, 3, 2, 10, 13, 8, 11 >$
4. $A =< 5, 3, 2, 6, 10, 13, 8, 11 >$

PART (ii)

$$\begin{aligned} T(n) &= T(0) + T(n-1) + \Theta(n) \\ &= \Theta(1) + T(n-1) + \Theta(n) \\ &= T(n-1) + \Theta(n) \\ &= \Theta(n^2) \end{aligned}$$

since

$$\sum_{k=1}^{n} \Theta(k) = \Theta(n^2)\,.$$

PART (iii)

$$\begin{aligned} T(n) &= 2\,T(\frac{n}{2}) + \Theta(n) \\ &= \Theta(n \log n) \end{aligned}$$

since it gives a recursion Tree with height $\log n$ and the sum of the costs of the nodes at each depth is $\Theta(n)$.

PART (iv)
We alternate lucky, unlucky, lucky, unlucky, lucky, ...

$$\begin{aligned} L(n) &= 2U(\frac{n}{2}) + \Theta(n) \qquad \text{lucky} \\ U(n) &= L(n-1) + \Theta(n) \qquad \text{unlucky} \end{aligned}$$

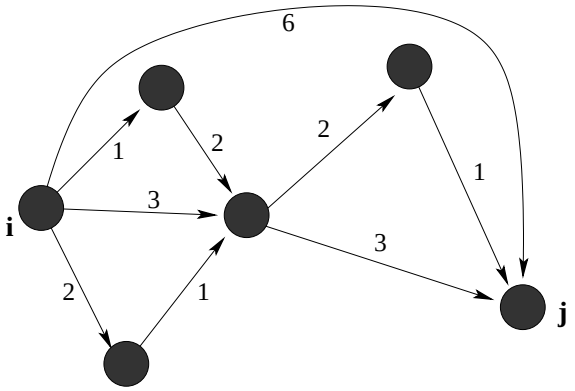
Solving it

$$\begin{aligned} L(n) &= 2(L(\frac{n}{2} - 1) + \Theta(\frac{n}{2})) + \Theta(n) \\ &= 2L(\frac{n}{2} - 1) + \Theta(n) \\ &= \Theta(n \log n) \end{aligned} \tag{1}$$

Questions 2

Let G be a directed graph with edges that have lengths. Present, in high-level pseudo-code, an efficient algorithm that computes, for **all** pairs of vertices i, j, the length of the shortest path from i to j, **and** the number of paths from i to j that have this shortest length. For this problem, the length of a path is the sum of its edge lengths. Assume that all cycles have positive total length. The edge lengths are stored in the array $E[1..n, 1..n]$, so that if (i, j) is an edge in G, then $E[i, j]$ is the length of the edge. You can also assume that $E(i, j)$ is infinite (i, j) is not in G.

Two paths are different if the sequence of edges that define the path are different. For example, in the figure below, there are seven different shortest paths from i to j.



Solution to Question 2

```
procedure Count(C[1..n,1..n],Pcount[1..n,1..n]);
  forall pairs i, j do
    if C[i, j] ≠ ∞ then
      Pcount[i, j] ← 1
    else
      Pcount[i, j] ← 0
    endif
  endfor;
  for k ← 1 to n do
    for i ← 1 to n do
      for j ← 1 to n do
        if C[i, j] > C[i, k] + C[k, j] then
          C[i, j] ← C[i, k] + C[k, j];
          Pcount[i, j] ← Pcount[i, k] × Pcount[k, j]
        elseif C[i, j] = C[i, k] + C[k, j] then
          Pcount[i, j] ← Pcount[i, j] + Pcount[i, k] timesPcount[k, j]
        endif
      endfor
    endfor
  endfor
end_Count.
```

Question 3

1) The Towers of Hanoi problem is the following.

You are given three posts, A, B, and C. There are n rings r_n, r_{n-1}, $\ldots$, r_2, r_1 sitting on post A, with r_i directly on top of r_{i+1}, for $1 \leq i < n$, so that ring r_1 is the top ring. The other posts are empty.

The problem is to move the rings so that they all wind up on post B subject to the following rules:

A ring r_i cannot be placed on a pole that holds a (smaller indexed) ring r_h where $h < i$.

Only the top ring on a post can be removed at any step.

Only one ring can be moved at each step, and it must be removed from one post and placed on another.

Recursion gives an easy solution to the problem.

```
procedure TH(n,A,B,C);
    if n = 1 then move top ring on A to B
    else
        TH(n − 1,A,C,B); { Move the top N − 1 rings form A to C }
        move the top ring on A to B;
        TH(n − 1,C,B,A); { Move the top N − 1 rings from C to B }
    endif
end-TH.
```

a) Present the recurrence equation for the exact number of ring moves for TH(n,A,B,C). Be sure to include the initial condition as well as the recurrence equation.

b) Now suppose that in addition to posts A, B and C, there is a short post D, which can hold a single ring of any type. All other aspects of the problem are unchanged. Present, in high level code, as efficient a solution to this new problem as you can,

c) Present the recurrence equation for the exact number of ring moves for the solution given in part b. Be sure to include the initial condition as well as the recurrence equation.

Solution to Question 3

a)

$T(1) = 1$;

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

b)

```
procedure THD(n,A,B,C);
    if n = 1 then move top ring on A to B
    elseif n = 2 then
        move top ring on A to D;
        move top ring on A to B;
        move top ring on D to B
    else
        TH(n − 2,A,C,B);
        THD(2,A,B,C,D);
        TH(n − 2,C,B,A)
    endif
end-THD.
```

c)

$T(1) = 1$;

$T(2) = 3$);

$T(n) = 2T(n-2) + 3$, $n > 2$.