Algorithm 중간고사

2000. 10.24 Open Book, 75분

* 문제를 풀기 위해 필요한 가정이 있으면 직접 가정하고 이를 명시할 것. 합리적인 가정도 실력이고, 쓸데없는 세부 사항에 집착하지 않는 것도 실력임.

1. (15) 다음의 asymptotic running time은? (Master's Theorem 사용할 수 있으면 해도 좋음)

   1.1 $T(n) = 3T(n/3) + n$
   1.2 $T(n) = 3T(n/2) + n\log n$
   1.3 $T(n) = 3T(n/3 + 5) + n/2$

2. (5) 바람직하지는 않지만 quicksort의 pivot을 찾기 위해 최대한 효율적인 selection algorithm을 이용하여 전체의 $n/4$번째 element를 먼저 찾은 다음 이 element를 pivot으로 사용한다고 하면 이 quicksort의 asymptotic running time은 어떻게 되는가?

3. (10) Red-black tree에서 root의 색깔이 현재 red이다. 이 상태에서 임의의 element가 insert되었는데 insert 이후에 이 RB tree의 black height가 하나 증가하였고 root node는 그대로 red color를 유지하고 있었다. 이런 경우가 발생할 수 있는가? 당신의 대답에 대해 justify하라.

4. (15) Uniform hashing을 가정할 때 unsuccessful search나 insertion에 필요한 probe 횟수의 기대치는 기껏해야 $\frac{1}{1-\alpha}$ 임을 수업 시간에 증명하였다. Uniform hashing은 만족시키기가 만만한 가정은 아니다. 그러면 만일 uniform hashing에 대한 조건을 없애고 $h_1(x), h_2(x), \cdots, h_m(x)$가 (0, 1, ..., m-1) 에서의 uniform distribution만을 만족한다고 할 때, unsuccessful search나 insertion의 probe 횟수의 기대치는 어떻게 되는가?

5. (15) n 개의 element를 가진 array $A[1 \ldots n]$에서, element x가 아래의 성질을 만족하면 *majority element*라 부른다.

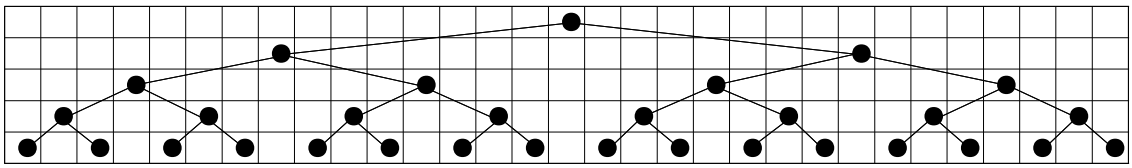
$$| \{ i | A[i] = x \} | > \frac{n}{2}$$

   집합 A의 element 수 n이 짝수일 경우, 집합 A에서 두 개씩 쌍으로 비교하여 일치하는 element만 남겨 새 집합 A'을 만든다. 즉,

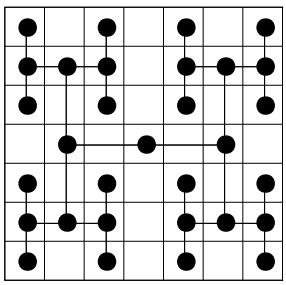
$$A' = \{ a_{2i},\ i = 1, \cdots, n/2 \mid a_{2i-1} = a_{2i} \}$$

   이 경우 element x가 A에서 majority element이면 x는 A'에서도 majority element임을 증명하라.

6. (20) n 개의 leaf node를 가진 임의의 complete binary tree를 그리는데는 root node를 맨 위에 두고, 그 children들을 바로 밑에 두는 방식으로 그릴 수 있다. 예를 들어, 아래는 n=16일 경우를 그린 거이다. 이 방법을 쓸 경우 tree를 그리는 데 필요한 asymptotic space는 $O(n \log n)$이다. (그림 전체를 포함할 수 있는 최소의 직사각형을 공간의 크기로 잡는다. 아래 예에서는 5*31 만큼의 space를 필요로 한다.)



이렇게 그리지 않고 complete binary tree를 H-tree라는 방식으로 그릴 수도 있다. n=16일 경우의 H-tree는 다음과 같다. H-tree로 n 개의 leaf node를 가진 complete binary tree를 그릴 때 필요한 asymptotic space의 크기는? Recurrence relation을 반드시 밝히고 이에 의거하여 추론할 것.



7. (20) 1부터 n까지 번호가 붙은 노드들이 있다. 모든 노드들간에는 직접 이동 할 수 있는 edge가 있고 (단, 번호가 큰 노드에서 작은 노드로는 이동할 수 없다) 이들 각각의 edge에는 고유의 점수가 배정되어 있다. 노드 1부터 시작해서 edge들을 따라 k 개의 노드를 방문할 때 총 점수의 가능한 최대값 t_k를 알아내는 알고리즘을 dynamic programming으로 작성하고 그 complexity를 분석하라.

오른쪽 그림의 예에서, 만일 점 1에서 시작하여 두 개의 노드를 방문하도록 지시 받았을 때(k=2) 총점을 최대화하는 방법은 1→2→4의 경로로 방문하는 것이고 총점은 12+11 = 23점이 된다. 반드시 마지막 노드에서 끝날 필요는 없다.