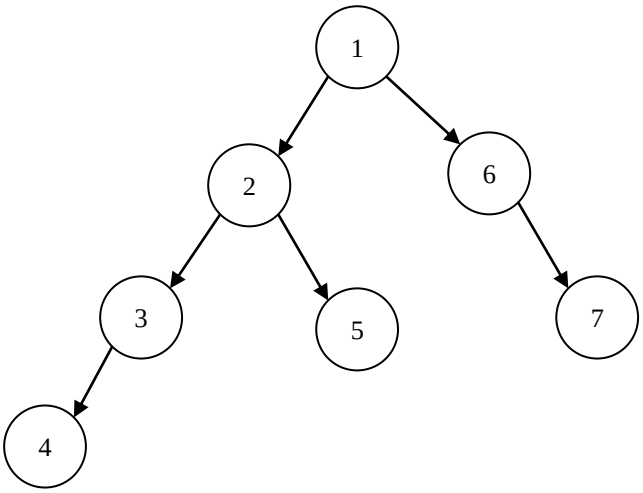
알고리즘 중간고사

2004. 10. 28, Open book, 75분

1. (15점) Asymptotic running time을 밝히라. (upper bound만 밝힐 것. 과정 포함.)
   A. (7점) $T(n) = 3T\left(\frac{n}{3}+1\right) + n$
   B. (8점) $T(n) = T\left(\frac{5n}{8}\right) + T\left(\frac{3n}{8}\right) + O(n)$

2. (15점) 아래는 directed graph로부터 만든 DFS tree이다. 노드의 번호는 DFS 과정에서 방문한 순서를 나타낸다. 이 DFS tree에서 존재할 수 있는 모든 가능한 cross edge의 수는 총 몇 개 인가? (총 수와 함께 아래 그림에서 cross edge들을 표시하라)

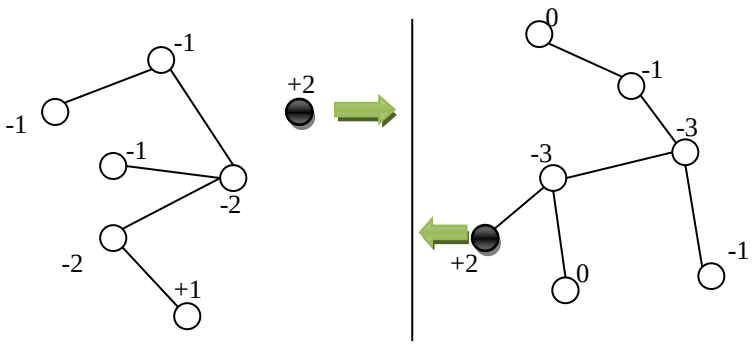


3. (15점) N 개의 원소를 sorting 하려고 하는데 각 원소들의 key는 $[0, N^3]$ 구간에서 확률적으로 고루 선택된 random integer들이다. 수업 시간에 배운 범위 내에서 당신이 생각하는 가장 실용적으로 효율적인 방법을 제시하고, 이의 running time을 채점자에게 설득시키라.

4. (15점) 수업 시간에 배운 worst-case linear time algorithm에서 5개로 나누는 대신 10개로 나누고, 이 10개짜리 그룹에서 i번째 작은 것들의 집합에서 median을 택하는 방식으로 수정하려 한다. 이 알고리즘이 linear time의 running time을 유지할 수 있는 최소의 i 값은 얼마인가?

5. (25점)
   A. (15점) 임의의 sequence에서 monotonically increasing subsequence란 item들의 key 값이 일관되게 증가하는 subsequence를 말한다. 임의의 sequence에서 longest monotonically increasing subsequence의 길이를 알아내는 $O(n^2)$의 dynamic programming 알고리즘을 제시하라. 문제의 간명함을 위해 모든 item의 key 값은 다르다고 가정한다.
   B. (10점) 이번에는 임의의 sequence에 있는 item들은 각각 두 개씩의 key 값 key1, key2를 갖는다. key1은 monotonically increasing, key2는 monotonically decreasing 하는 longest subsequence를 찾는 $O(n^2)$의 알고리즘을 제시하라. (질문 불허. 불확실한 것이 있으면 스스로 가정하고 할 것.)

6. (20점) Undirected graph $G = (V, E)$에서 graph의 vertex들을 겹치지 않고 두 개의 같은 크기 (vertex의 총 수가 홀수이면 1 차이가 난다.)의 집합 (A, B)로 나눈 것을 "graph bisection"이라 한다. 임의의 bisection이 주어졌을 때, edge (i, j)에서 vertex i와 vertex j가 각각 다른 쪽 집합에 속해 있으면, edge (i, j)는 cross edge라 부른다. 임의의 graph bisection이 갖는 cross edge의 총 수를 cut size라 부른다. 특정한 bisection 상에서 vertex v 하나만 반대쪽으로 옮겨가면 (물론 일시적으로 균형이 살짝 깨진다) cut size가 얼마나 감소하는지를 v의 gain이라 한다. 아래 그림은 graph bisection과 gain을 예로 보여준다.



cut size = 6

7. 주어진 bisection (A, B)로부터 cut size가 더 작은 bisection을 찾기 위한 알고리즘 Pseudo-KL은 다음과 같이 작동한다.

A. 모든 vertex에 대한 gain을 계산한다.
B. 집합 A에서 gain이 가장 큰 vertex u와 집합 B에서 gain이 가장 큰 정점 v를 서로 교환해서 새 bisection을 얻는다. Gain이 가장 큰 것이 두 개 이상 있을 때에는 index가 작은 vertex를 선택한다. 이제 vertex u와 vertex v는 고정시키고 더 이상 교환의 대상이 되지 않는다.
C. b의 과정을 더 이상 교환할 vertex가 남지 않을 때까지 $\frac{|V|}{2}$번 반복한다. 물론 이 과정에서 최대 gain이 음수가 되는 경우도 발생할 수 있다.
D. c의 과정에서 교환된 vertex들을 순서대로 $< u_1, u_2, \cdots, u_{\frac{|V|}{2}} >$, $< v_1, v_2, \cdots, v_{\frac{|V|}{2}} >$라 할 때 gain의 총합 (gain sum)이 가장 큰 $< u_1, u_2, \cdots, u_i >$, $< v_1, v_2, \cdots, v_i >$를 찾아 이들까지만 교환하고 이후의 교환은 없었던 것으로 한다. 최대 gain sum이 0 미만이면 아무런 교환을 행하지 않는다. 최대 gain sum을 가진 $< u_1, u_2, \cdots, u_i >$, $< v_1, v_2, \cdots, v_i >$ 가 둘 이상이면 가장 긴 것(원래의 그래프에서 가장 많이 변화시킨 것)을 택한다.

위의 알고리즘을 가장 효율적으로 구현하면 running time이 어떻게 될 것 같은가? 이 때 사용하는 자료구조를 반드시 설명해야 함. (지나치게 복합한 설명은 사절)