Intro to Algorithm Assignment #3 20181257 최관철.

1. (1 point) Prove that equation $T(n) = 2^n$ is true, given the equation $T(n) = 1 + \sum_{j=0}^{n-1} T(j)$ and the initial condition $T(0) = 1$ of the rod-cutting problem. Use inductive proof for your solution.

Hint: $\sum_{k=0}^{n} x^k = \frac{x^{n+1}-1}{x-1}$

Proof by induction.

$$\left.\begin{array}{l} T(n) = 2^n, \quad T(0) = 1 \\ T(n) = 1 + \sum_{j=0}^{n-1} T(j) \quad \text{for } \forall n \geq 1 \end{array}\right\} \cdots \text{Inductive Hypothesis. (I.H)}$$

Base case : For $n=1$, $T(1) = 1 + T(0)$

$$= 1 + 1 = 2^1 \quad \text{True!}$$

Inductive step : Assume that $T(n) = 1 + \sum_{j=0}^{n-1} T(j)$

$$= 1 + \sum_{j=0}^{n-1} 2^j \quad \text{is true.}$$

Then, we need to show $T(n+1)$ is also true.

$$T(n+1) = 1 + \sum_{j=0}^{n} T(j)$$

$$= 1 + \sum_{j=0}^{n} 2^j \quad \cdots \text{I.H.}$$

$$= 1 + \frac{2^{n+1}-1}{2-1}$$

$$= 2^{n+1} \quad \text{True!}$$

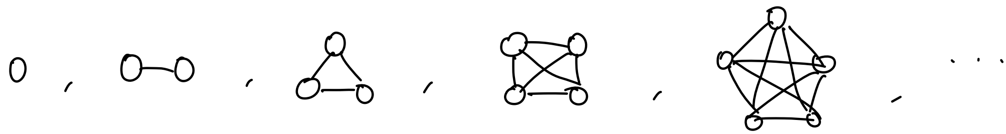
Q.E.D.

2. Answer the following questions, including brief descriptions of how you get your answers.

- (1 point) What is the maximum number of edges in an undirected graph with V vertices and no parallel edges? Parallel edges (also called multiple edges or a multi-edge) are, in an undirected graph, two or more edges that are incident to the same two vertices. ··· (a)

- (1 point) What is the minimum number of edges in an undirected graph with V vertices, none of which are isolated? ··· (b)

G :



$|V|$: 1 , 2 , 3 , 4 , 5 , ...

$|E|$: 0 , 1 , 3 , 6 , 10 , ...

(a) Maximum numbers of edge mean that all vertices are connected with adjacent edges exhausively. In other words, a vertex is connected to every other vertices except itself : there are $n-1$ edges ($n = |V|$). This applies to all vertices (i.e., $n \times (n-1)$) which will result in duplicate edges, so we need to consider it.

$\therefore$ Maximum numbers of edges for n vertices $= \underline{\underline{\dfrac{n \times (n-1)}{2}}}$

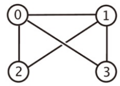
(b) Minimum numbers of edges mean that there only needs to be one edge between vertices.

$\therefore$ Minimum numbers of edges for n vertices $= \underline{\underline{n-1}}$

3. (1 point) Fill out ①, ②, ③, and ④ in the below table.

①, ③: Write an answer to the instance of the problem given in the "Instance" column.

②, ④: Select which category ((P, NP, NP-hard, or NP-complete) the given problem belongs.

	Problem	Instance	Solution to instance	P/NP/NP-hard/NP-complete
1	Finding a simple path that visits every vertex exactly once	 (starting from node 0)	①	②
2	Assign values to variables, each of which has two possible values (TRUE or FALSE), in order to satisfy a system of constraints on pairs of variables	$(x_1 \vee x_2) \wedge (x_2 \vee \bar{x}_1) \wedge (\bar{x}_1 \vee \bar{x}_2)$	③	④

① : $\langle 0, 2, 1, 3 \rangle$ or $\langle 0, 3, 1, 2 \rangle$

② : NP - Complete

③ : $\langle x_1 =$ FALSE, $x_2 =$ TRUE $\rangle$

$$\Rightarrow (x_1 \vee x_2) \wedge (x_2 \vee \bar{x}_1) \wedge (\bar{x}_1 \vee \bar{x}_2)$$

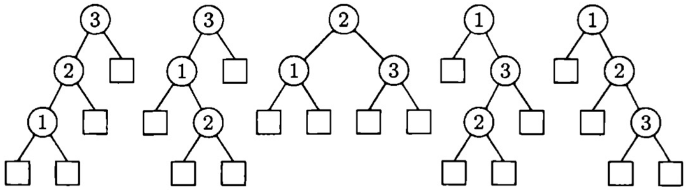
$$= (F \vee T) \wedge (T \vee T) \wedge (T \vee F)$$

$$= T \wedge T \wedge T = T.$$

④ : NP - Complete

4. (1 point) Let's assume that there is a binary search tree having three nodes, and their keys $K_1 < K_2 < K_3$ have respective probabilities of 0.2, 0.5, and 0.3 for searching (i.e., we search for the first, second, and third node of the probabilities of 0.2, 0.5, and 0.3). What is the expected number of comparisons required to search for a node?

Hint: Given three nodes with their keys $K_1 < K_2 < K_3$, there are five possible trees as shown below (by assuming they are equally likely). In the figure, the circles denote each node, and the rectangles denote NIL. For example, when we search for node 1 in the first tree, the number of comparisons required to find it is 3. Similarly, when we search for node 3 in the fourth tree, the number of comparisons is 2.



(a) (b) (c) (d) (e)

* Expected number of comparisons $= \sum_{i \in \{Node\}}$ (number of comparisons for i) $\times$ (search probability for i)

(a). E(number of comparisons) $= 3 \times 0.2 + 2 \times 0.5 + 1 \times 0.3$

$= 1.9$ (computed in order by K_i for $1 \le i \le 3$)

(b). " $= 2 \times 0.2 + 3 \times 0.5 + 1 \times 0.3$

$= 2.2$

(c) " $= 2 \times 0.2 + 1 \times 0.5 + 2 \times 0.3$

$= 1.5$

(d) " $= 1 \times 0.2 + 3 \times 0.5 + 2 \times 0.3$

$= 2.3$

(e) " $= 1 \times 0.2 + 2 \times 0.5 + 3 \times 0.3$

$= 2.1$

5. (1 point) Let's consider a scenario where we need to arrange a set of classes (activities) in numerous classrooms. Suppose 1) a class has an arbitrary start time and finish time, 2) each class can be held in any classroom, and 3) only one class can be held in a classroom at a time. Our goal is to schedule all the classes while minimizing the number of classrooms required. Derive a greedy algorithm that determines the optimal assignment of classes to classrooms in such a way that the total number of needed classrooms is minimized.

Let a set of start time and finish time of each class be s and f, where are sorted by an increasing order of start time ($s_0 \le s_1 \le \cdots \le s_{n-1}$) : $O(n \log n)$.

Let each classes (activities) be a_i for $0 \le i \le n-1$.

```
def ClassRoomsNeeded (s, f) :
    n = s.length
    A = [[a_i]]
    for i = 1 to n-1 do
        Allocate_Flag = 0
        for j = 0 to A.size - 1 do
            if s[i] < A[j][-1].FinishTime then   // compare new start
                Allocate_Flag += 1                 time with last finish
            else                                   time of each allocated
                A[j].insert(a_i)                   classroom.
                break
        if Allocate_Flag == A.size then
            A.insert([a_i])
    return A          // minimum (optimal) needed classrooms
```

6. Answer the following questions, including brief descriptions of how you get your answers.

- (1 point) Assume we have two problems that are known to be NP-complete. Does this mean that there is a polynomial-time reduction between them? ... (a)
- (1 point) Let's assume that X is NP-complete, X polynomial-time reduces to Y, and Y polynomial-time reduces to X. Then, is Y necessarily NP-complete? ... (b)

(a) ~~A~~ both problem A and problem B are known as NP-complete, it means that there exists polynomial-time reduction between them. Since NP-complete problems are themselves NP problems, all NP-complete problems can be reduced to each other in polynomial time.

(b). To show that Y is complete, we need the followings:

① Y is in NP

② Y is NP-Hard

X is NP-complete, which means it's polynomial time many-to-one reducible. Therefore, if Y polynomial time reduces to X, then Y is in NP ... ①

And if X polynomial time reduces to Y, then Y is NP-hard ... ②

∴ Y is necessarily NP-complete.