중간고사 1

1. 다음을 간단히 설명하시오.
(a) Time complexity
(b) Time complexity에서 best case 보다 worst case가 더 중요한 이유
(c) Time complexity 표현에 굳이 점근적 표현을 적용하는 이유
(d) DNA sequence alignment 문제
(e) Optimal binary search tree 문제

2. 다음의 경우에 가장 정확한 점근적 바운드 표현은? 표현할 수 없는 경우는 X 로 표기 (Ta: average case; Tw: worst case, Tb: best case)
(a) $n \log n + 10n \leq T(n) \leq 2^n + 5n^2$

	T(n)	Tb(n)	Ta(n)	Tw(n)
O 표현				
Ω 표현				
Θ 표현				

(b) $n + 1 \leq Tb(n) \leq 2n + 3$

	T(n)	Tb(n)	Ta(n)	Tw(n)
O 표현				
Ω 표현				
Θ 표현				

(c) $3n + \log n \leq Tw(n) \leq n^2 + 2\, n \log n$

	T(n)	Tb(n)	Ta(n)	Tw(n)
O 표현				
Ω 표현				
Θ 표현				

3. Dynamic programming 알고리즘과 관련하여 다음을 간단히 설명하시오.
(a) 일반적인 구조
(b) divide and conquer 알고리즘과 비교
(c) 최적성의 원리가 필요한 이유
(d) divide and conquer 알고리즘에서는 최적성의 원리가 필요하지 않은 이유

4. 아주 큰 n 자리 정수를 곱하는 문제에 대한 교재의 알고리즘에 대하여
(a) 구조를 간단히 설명하시오.
(b) 이것이 devide and conquer 알고리즘인 근거를 설명하시오.
(c) 종이에 적는 일반적인 정수 곱셈 알고리즘은 시간복잡도가 $\Theta(n^2)$임을 설명하시오.
(b) 교재의 알고리즘은 시간복잡도가 (c)에 비해서 개선되는 근거를 설명하시오.

5. 모든 쌍의 vertex 간의 최단 거리를 구하는 문제에 대하여
(a) 교재의 Floyd 알고리즘이 Dynamic programming 알고리즘이라고 하는 근거를 설명하시오
(b) 이 문제에 divide and conquer 알고리즘을 사용하지 않고 이렇게 dynamic programming 알고리즘을 적용하면 시간복잡도가 개선되는 근거는 무엇인가? (구체적인 예를 들어야 함)
(c) Divide and conquer에 비해서 얼마나 개선되는지를 설명하시오.

중간고사 2

1. 다음을 설명하시오.
(a) Huffman code 문제
(b) 상태공간 트리를 스케줄링 문제를 예로 들어서 설명
(c) Depth first search를 Traveling salesperson 문제를 예로 들어서 설명
(d) Monte Carlo estimation을 graph coloring 문제를 예로 들어서 설명

2.Greedy approach와 Dynamic programming 알고리즘을 비교하시오.
(a) 형식에 있어서 비슷한 점
(b) 알고리즘 진행과정의 차이점
(c) 알고리즘의 정확성을 하나는 증명할 필요가 없고 다른 하나는 증명이 필요한 이유

3. Backtracking과 Branch and bound 알고리즘을 비교하시오. (0/1 Knapsack 문제를 예시로 활용)
(a) 형식에 있어서 차이점
(b) 알고리즘 진행과정의 비슷한 점
(c) 알고리즘 진행과정의 차이점

4. Traveling salesperson 문제를 위해서 다음을 알고리즘 형식으로 작성하시오.
(a) Backtracking을 적용한 기본구조
(b) Backtracking에 적용 가능한 promising 함수
(c) Branch and bound를 적용한 기본구조
(d) Branch and bound에 적용 가능한 bound 함수

기말고사

1. 다음을 설명하시오.
(a) P 문제
(b) NP 문제
(c) $A \propto B$ 라는 의미
(d) $A \propto_T B$ 라는 의미
(e) Pseudo polynomial time 알고리즘

2. 계산복잡도와 관련하여
(a) NP-Complete 문제의 정의
(b) 0/1 Knapsack Decision 문제가 NP-Complete 라는 의미
(c) NP-Hard 문제의 정의
(d) 위의 (b)를 전제로 0/1 Knapsack 문제가 NP-Hard 라는 것의 증명

3. Sorting 문제와 관련하여

(a) Sorting 문제는 계산복잡도가 $\Omega(n \log n)$ 이라는 의미

(b) Merge sort 알고리즘의 최악 시간복잡도가 $O(n \log n)$ 이라는 의미

(c) 위의 (a)와 (b)를 결합하면 도출할 수 있는 결론

(d) Radix sort 알고리즘은 시간복잡도가 $\Theta(n)$인데 위의 (a)를 위배할 수 있는 이유