

CS341 Final F2005

1. [Not covered]
- A) True. $2^{n+3} \leq c \cdot 2^n$ for $c \geq 8$
 - B) True. $n \leq n \log n$ for $n \geq 2$
 - C) True.
 - D) True.
 - E) False.

- 2.
- A) False.
 - B) False [not covered]
 - C) False?
 - D) True
 - E) False [not covered]

3. [Not on final]
- $T(n) = 3T(n/2) + O(n^{1+e})$
- convert(B, n)
- if (n == 1) return B[0]
 - upper = B[n/2 to n-1]
 - lower = B[0 to n/2 - 1]
 - multiplier = $[2^{(n/2-1)}]$ as binary number
 - lowerDec = convert(lower, n/2)
 - upperDec = convert(upper, n/2)
 - multiplierDec = convert(multiplier, n/2-1)
 - return (upperDec * (multiplierDec + multiplierDec)) + lowerDec

- 4.
- A) Floyd-Warshall
 - B) There isn't a shortest path between any pair of vertices which form part of a negative cycle since you can loop infinitely through the negative cycle to decrease the cost of the path.

C) Run Floyd-Warshall, and check if the distance of any vertex to itself is negative.

5. A) We first prove the problem is in NP. Given two subsets A' and B' , it takes polynomial-time to check that A' is a valid subset of A and B' is a valid subset of B . It takes polynomial-time to check that the sum of A' is equal to the sum of B' . Hence we have found a polynomial-time verification algorithm for this problem.

We prove that this problem is a reduction of subset-sum. Given an instance of subset sum with positive integers $a_1 \dots a_n$ and target value K , we create an instance of our problem. Set A is the set of numbers $a_1 \dots a_n$. $B = \{K\}$: it is a set with a singular number K .

If there exists a solution to this instance of subset sum, then we have that there is a subset S of the numbers where $\text{sum } S = K$. Then since $A = a_1 \dots a_n$, S is a subset of A , which sums to K . Then the subset $\{K\}$ of B also sums to K , so we have found subsets of A and B that sum to K , so have an equal sum.

If there exists a solution to this instance of our problem, we have that the only non-empty set of B is $\{K\}$. Then there must be a subset of A that sums to K , so we have a solution to our instance of subset sum.

It takes linear time to construct the instance of our problem from subset sum. Hence our problem is a polynomial-time reduction of subset sum. Since subset sum is NP-complete, our problem is also NP-complete.

B) A set of size k has 2^k subsets. So there are 2^n subsets from A and 2^m subsets from B . It takes $2^n \cdot 2^m$ time to compare all pairs. Then a brute force algorithm would have runtime $O(2^{n+m})$.

C) [Not covered]

6. [Not covered]