Problem 1: Dynamic Programming, Max Sum Contiguous Subsequence (10 points)
A contiguous subsequence of a list S is a subsequence made up of consecutive elements of S. For instance, if S is $(5, 15, -30, 10, -5, 40, 10)$ then $(15, -30, 10)$ is a contiguous subsequence, but $(5, 15, 40)$ is not. Give a linear time algorithm for the following task:
Input: A list of numbers, $(a_1, a_2, \ldots, a_n)$.
Output: The contiguous subsequence of maximum sum (a subsequence of length zero has sum zero).
For the preceding example, the answer would be $(10, -5, 40, 10)$, with a sum of 55.
Hint: For each $j \in \{1, 2, \ldots, n\}$ consider a contiguous subsequences ending exactly at position j.

Answer:

Problem 2, Dynamic Progreamming Application (10 points)
You are going on a long trip. You start on the road at mile post 0. Along the way there are n hotels, at mile posts $a_1 < a_2 < \ldots < a_n$, where each a_i is measured from the starting point. The only places you are allowed to stop are at these hotels, but you can choose which of the hotels you stop at. You must stop at the final hotel (at distance a_n), which is your destination. You would ideally like to travel 200 miles a day, but this may not be possible (depending on the spacing of the hotels). If you travel x miles during a day, the penalty for that day is $(200 - x)^2$. You want to plan your trip so as to minimize the total penaltythat is, the sum, over all travel days, of the daily penalties. Give and analyze a polynomial time algorithm that determines the optimal sequence of hotels at which to stop.

Answer:

Problem 3, Dynamic Programming, Making Change (10 points)
Given an unlimited supply of coins of denominations $x_1 < x_2 < \ldots < x_n$, we wish to make change for a given value v; that is, we want to find a set of coins whose total value is v. This might not be possible: for example, if the denominations are 5 and 10 then we can make change for 15 but not for 12. Give an $O(nv)$ dynamic programming algorithm for the following problem:
Input: $x_1, x_2, \ldots, x_n$
Output: Yes or No, is it possible to make change for v using denominations $x_1, x_2, \ldots, x_n$?

Answer:

Problem 4, Dynamic Programming, Making Change Optimally (10 points)
Here is yet another variation on the change-making problem. Given an unlimited supply of coins of denominations $1 = x_1 < x_2 < \ldots < x_n$, we wish to make change for a value v using as few coins as possible (note that, since $x_1 = 1$, any value v is realizable.)
(a) Give an $O(nv)$ algorithm that finds the minimum number of coins, say $k(v)$, whose total value is v.
(b) Give an $O(nv)$ algorithm that finds a solution $s_1, \ldots, s_n$, where s_i denotes the number of coins of denomination x_i, $1 \leq i \leq n$, such that (a)$\sum_{i=1}^{n} s_i x_i = v$ and (b)$\sum_{i=1}^{n} s_i = k(v)$.

Answer:

Problem 5, Dynamic Programming, Longest Path out of Specific Vertex (10 points)
Let $G(V, E)$ be a directed acyclic graph presented in topologically sorted oredr, that is, if $V = \{v_1, \ldots, v_n\}$, then $(v_i \to v_j \in E) \Rightarrow (i < j)$ - ie, all edges are directed from lower order vertices to higher order vertices. Give an $O(|V| + |E|)$ complexity alogithm that finds the length of the longest path that starts in v_1.

Answer:

Problem 6, Wiggly Sorting, KSelect Application (10 points)
Let $a_1, \ldots, a_n$ be an unsorted array of n distinct integers, where n is odd. We say that the array is *wiggly-sorted* if and only if $a_1 < a_2 > a_3 < a_4 > a_5 < \ldots > a_{n-2} < a_{n-1} > a_n$.
Give a linear time algorithm that wiggly-sorts the initial array $a_1, \ldots, a_n$.

Answer:

Problem 7, Finding Median under Partial Sorting (10 points)
Let X and Y be sorted arrays of length n. Suppose also that the elements of $X \cup Y$ are all distinct. Give an $O(\log n)$ comparison algorithm that finds the n-th element of $X \cup Y$.

Answer:

Problem 8, Strong Majority, KSelect Application (10 points)
Let $a = a_1 \dots a_n$ be an input array of n unsorted and not necessarily distinct numbers. We say that a value m is a *strong majority* of a if and only if the value of at least $\left(\lfloor \frac{n}{2} \rfloor + 1\right)$ elements of a is equal to m:

$$|\{i : a_i = m, 1 \leq i \leq m\}| \geq \left(\lfloor \frac{n}{2} \rfloor + 1\right) \quad .$$

Give a linear time algorithm that determines if the input array a has a strong majority.
If the answer is YES, then give also the value m of the strong majority.

Answer:

Problem 9, Unknown Length Binary Search (10 points)
You are given an infinite array $a(\cdot)$ in which the first n cells contain positive integers in sorted order and the rest of the cells are not defined. By not defined, we mean that if an algorithm probes a position $i > n$ then the program returns an error message. You are not given the value of n. Describe an $O(\log n)$ comparison algorithm that takes a positive integer x as input and finds a position in the array containing x, if such a position exists, otherwise the algorithm returns NO.

Answer:

Problem 10, Counting Significant Inversions (10 points)
Let $\overline{a} = (a_1, \ldots, a_n)$ be an array of n distinct unsorted numbers. Say that a pair $1 \leq i < j \leq n$ is a *significant inversion* on $\overline{a}$ if and only if $a_i > 2a_j$. Give an $O(n \log n)$ comparison algorithms that determines the number of inversions of an arbitrary array of n distinct unsorted integers.
Answer: