When asking for designing an algorithm, both English description and pseudo-code are required. When asking for an efficient algorithm, please analyze the complexity of your algorithm; the score will depend on how efficient your algorithm is.

1. (30 points) Below is a weighted undirected graph. (a) Please list the vertices in the order of adding to MST (minimum spanning tree) by Prim’s algorithm, starting with vertex A (ties are broken by alphabet order). Please also list the MST. (b) Let’s add more edges to this graph so that there is an edge between any two vertices in the new graph and if an edge is not in the original graph, then its weight is equal to the minimal distance between these two vertices in the original graph. Does this new graph satisfy the triangle inequality? If not, please point out the violation; if yes, please use the MST in (a) to generate a Traveling Salesman Tour and compute the cost of the tour.

Answer: (a) The vertex list is: A, F, B, C, H, D, E, G. The MST consists of the following edges: (A, F), (A, B), (B, C), (C, H), (B, D), (F, E), (F, G).

(b) The triangle CEH does not satisfy the triangle inequality: w(CE)+w(CH) = 3+1 < 6 = w(EH).

2. (30 points) Let S be an array of n unsorted numbers. We want to select a number x from S such that x is less than or equal to the median of S. (a) Please design an efficient algorithm to solve this problem. (b) If we are allowed to select x larger than the median of S with probability p < 0.001, please design an efficient randomized algorithm to solve the same problem.

Answer: (a) We search for the minimal number in the first n/2 numbers that will be less than the median of S. Assume S is stored in array S and has n elements. The pseudo code is as follows:

```plaintext
Min = S[0]
for (j=1; j<= n/2; j++) if (Min > S[j]) Min = S[j];
return Min;
```

The complexity is O(n).

(b) If we randomly pick a number x in S, the probability that x > median(S) is less than 1/2. If we randomly pick k numbers in S, the probability that all these numbers > median(S) is less than 1/2^k. When k=10, 1/2^k =1/1024 < 0.001. From this analysis, we can design an randomized algorithm as follows: If n <= 20, use the algorithm in problem (a); otherwise, assume random(n) generates a random number between 0 and (n-1),

```plaintext
k = random(n); Min = S[k];
for (j=1; j<= 9; j++) { k = random(n); if (Min > S[k]) Min = S[k];
return Min;
```
If the returned \( \text{Min} > \text{median}(S) \), then all 10 selected numbers are > median(S). From the above analysis, this probability is < 0.001. The complexity of this algorithm is \( O(1) \).

3. (40 points) Given a list \( L \) of \( n \) positive numbers, let \( \text{max}(L) \), \( \text{min}(L) \), \( \text{sum}(L) \) be the maximal number, minimal number, the summation of all numbers in \( L \), respectively. The \textbf{partition} problem is to divide \( L \) into two parts, \( L_1 \) and \( L_2 \), such that \( \text{sum}(L_1) = \text{sum}(L_2) \). (a) Please formulate the partition problem as a CSP (constraint satisfaction problem).  (b) An approximate algorithm to the partition problem is to use a greedy strategy: At first, sort \( L \) in non-increasing order. Suppose \( L_1 \) and \( L_2 \) are empty initially. For each number \( x \) in \( L \) from \( \text{max}(L) \) to \( \text{min}(L) \), if \( \text{sum}(L_1) > \text{sum}(L_2) \), add \( x \) to \( L_2 \); otherwise add \( x \) to \( L_1 \). Please show that this algorithm cannot guarantee \(|\text{sum}(L_1) – \text{sum}(L_2)| <= \text{min}(L)|\). (c) Please design an \( O(n^2 \text{max}(L)) \) time algorithm for the partition problem.

Answer: The partition problem is a special case of the subset sum problem (when the subset sum is \( \text{sum}(L)/2 \)), which is a special case of the knapsack problem.

(a) Let \( L = (a_1, a_2, \ldots, a_n) \). The CSP for the partition problem consists of \( n \) variables \( (x_1, x_2, \ldots, x_n) \), each \( x_i \) taking 0 or 1. If \( x_i = 0 \), it means \( a_i \) is in \( L_1 \); otherwise in \( L_2 \). The constraint is \( a_1x_1 + a_2x_2 + \ldots + a_nx_n = \text{sum}(L)/2 \).

(b) Let \( L = (7, 6, 4, 4, 4, 1) \). The only solution is \( S_1 = (7, 6) \) and \( S_2 = (4, 4, 4, 1) \). The approximate solution is \( L_1 = (7, 4, 1) \) and \( L_2 = (6, 4, 4) \). \(|\text{sum}(L_1) – \text{sum}(L_2)| = 2 > 1 = \text{min}(L)|\).

(c) Like the Subset Sum problem, we may use dynamic programming to solve this problem. We first compute \( \text{sum}(L) \). If \( \text{sum}(L) \) is odd, then the answer is no (assuming all the numbers are integers). If it’s even, let \( K = \text{sum}(L)/2 \). Then the partition problem returns yes iff the Subset Sum problem returns yes for \((L, K)\). Suppose \( L \) has \( n \) numbers, let \( \text{decide}(i, k) = true \) iff there is a subset \( X \) from the first \( i \) numbers of \( L \) such that \( \text{sum}(X) = k \). Then

\[
\text{decide}(i, 0) = \text{true} \text{ for any } i; \\
\text{decide}(0, k) = \text{false} \text{ for any } k > 0; \\
\text{decide}(i, k) = \text{decide}(i-1, k) \ || \text{decide}(i-1, k – a_i) \text{ for any } i>0 \text{ and } k > 0;
\]

The order to compute \( \text{decide}(i, k) \) will be from small to large. The pseudo code to compute \( \text{decide}(n, K) \) is as follows:

\[
\text{for } (i = 0; i <= n; i++) \ \text{D}[i, 0] = \text{true}; \\
\text{for } (k = 1; k <= K; k++) \ \text{D}[0, k] = \text{false}; \\
\text{for } (i = 1; i <= n; i++) \\
\quad \text{for } (k = 1; k <= K; k++) \\
\quad \quad \text{if } (\text{D}[i-1, k] \ || \ (a_i <= k \ \&\& \ \text{D}[i-1, k – a_i])) \ \text{D}[i, k] = \text{true}; \\
\text{return D}[n, K];
\]

The complexity is \( O(nK) \). Since \( K <= n \times \text{max}(L) \), so it’s solvable in \( O(n^2 \text{max}(L)) \).