Final Exam CS 600

Name: Lan Chang

Question 1

Answer:

For this algorithm, we can use a balanced binary tree. For the external nodes in the tree, we can store all the points in set S in the. And we can store these points ordered by their keys. For the internal nodes in the tree, we can store how many external nodes are in the subtree of this internal node. When we want to find all the items with ranks in the range [a, b], we can use the 1DTreeRangeSearch algorithm.

Running time:

For this algorithm, the total running time is $O(\log n + k)$, which n is the number of the nodes and k is the number of answers.

Question 2

Answer:

For this algorithm, we can use Dijkstra's Algorithm. For each channel that is not eavesdropped by the hacker, we can add value 1 for this edge, and for each channel that is eavesdropped by the hacker, we can add value m for this edge. For using this algorithm, we can easily find a path from headquarters node s to one of its field nodes t. If the total value for this path is smaller than m, it means that there is not any channel that is eavesdropped by the hacker. And if it is bigger than m, it means that there is at least one channel that is eavesdropped by the hacker. I choose to use m as the value of the channel eavesdropped because if we use the whole edges without any channel eavesdropped to make up this path, the total value is not bigger than m and using Dijkstra's Algorithm is efficient.

Running time:

For this algorithm, the total running time is $O(m \log n)$, which m is the number of edge and n is the number of node.

Question 3

Answer:

If all edge capacities are bounded by a constant c, the maximum flow for this graph must be bounded by $O(c \times (m-1)) = O(m)$. As a result, the total running time in worst-case for this algorithm is $O(m^2)$, which m is the number of edge.

Question 4

Answer:

For each web page, we can store them to a data structure with inverted files. For the inverted files, we can store the key-value pairs (w, L), where w is a word and L is a collection of references to pages containing word w. For detecting whether a web page of length n has been previously encountered, we can compare this page with this data structure if they have the same inverted files, we need to compare n words, as a result, the total running time for detecting is $O(n)$. If there is not an inverted file could match this page, we need to add the inverted file of this page, as a result, the total running time for adding is $O(1)$. For this data structure, we can use the trie, for each node, we can store the single word, the parent of this node is the previous word of this word in the page, and the child of this node is the next word of this word in the page. And for the last node, we can store the inverted file representing the web page. Finally, we can see that this trie is stored ordered by the sequence of the words in the page.

Question 5

Answer:

We can use branch-and-bound algorithm to solve this problem. In this algorithm, we need to find the best solution at each shelf. At each shelf, this algorithm will find the best solution.

Pseudo-code:

Algorithm ShelfPack(S):

Input: Set S of n items which each item's size is s_i.

Output: The minimum number of shelves necessary to pack all the items.

$F \leftarrow \{(x, \emptyset)\}$ {Frontier set of subproblem configurations}
$b \leftarrow (+\infty, \emptyset)$ {Cost and configuration of current best solution}

while $F \neq \emptyset$ **do**
  select from F the most "promising" configuration (x,y)
  expand (x,y), yielding new configurations $(x_1,y_1), \ldots, (x_k,y_k)$
  for each new configuration (x_i,y_i) **do**
    perform a simple consistency check on (x_i,y_i)
    if the check returns "solution found" **then**
      if the cost c of the solution for (x_i,y_i) beats b **then**
        $b \leftarrow (c, (x_i,y_i))$
      else
        discard the configuration (x_i,y_i)
    if the check returns "dead end" **then**
      discard the configuration (x_i,y_i) {Backtrack}
    else
      if $lb(x_i,y_i)$ is less than the cost of b **then**
        $F \leftarrow F \cup \{(x_i,y_i)\}$ {(x_i,y_i) starts a promising search path}
      else
        discard the configuration (x_i,y_i) {A "bound" prune}
return b

Question 6

Answer:

For this algorithm, we can divide the set X into two sets X1 and X2, which each set's size is n/2. And we can recursively construct the polynomials p1(n) and p2(n) for these two sets, and for these two polynomials, p1(X1)=0 and p2(X2)=0. Then, we can use Fast Fourier Transform to compute the result of $p_1 \cdot p_2$.

Running time:

For this algorithm, the recurrence is $\mathrm{T}(n) = 2T(n/2) + bn \log n$. As a result, the total running time is $\mathrm{O}(n\log^2 n)$.