Homework 3

Algorithm Design

- 4-1 Decide whether you think the following statement is true or false. If it is true, give a short explanation. If it is false, give a counterexample.
    - Let G be an arbitrary connected, undirected graph with a distinct cost c(e) on every edge e. Suppose e* is the cheapest edge in G; that is, c(e*) < c(e) for every edge e $\neq$ e* . Then there is a minimum spanning tree T of G that contains the edge e*.

  Solution

  It is true. e* is the first edge that would be considered by Kruskal's algorithm, and so it will be included in the minimum spanning tree.

- 4-8 Suppose you are given a connected graph G, with edge costs that are all distinct. Prove that G has a unique minimum spanning tree.

  Solution

  假设T和T‘都两个图G的最小生成树。因为T和T'拥有相同的边数，但costs不相同，所以存在一个e’属于T而不属于T‘。若把e’加入到T‘中，就会得到一个cycle C。让e成为cycle C中最大的边，根据Cycle的性质，e应该不属于任何最小生成树，这与命题矛盾。得证

- 4-9 One of the basic motivations behind the Minimum Spanning Tree Problem is the goal of designing a spanning network for a set of nodes with minimum total cost. Here we explore another type of objective: designing a spanning network for which the most expensive edge is as cheap as possible.

  Specifically, let G = (V , E) be a connected graph with n vertices, m edges, and positive edge costs that you may assume are all distinct. Let T = (V, E') be a spanning tree of G; we define the bottleneck edge of T to be the edge of T with the greatest cost.

  A spanning tree T of G is a minimum-bottleneck spanning tree if there is no spanning tree T' of G with a cheaper bottleneck edge.

  (a) Is every minimum-bottleneck tree of G a minimum spanning tree of G? Prove or give a counterexample.
  (b) Is every minimum spanning tree of G a minimum-bottleneck tree of G? Prove or give a counterexample.

  Solution

  (a) False. 设在图G中有V_1, V_2, V_3, V_4 四个顶点，每两个顶点间有一个边，边V_i到V_j的权重为 i + j . 由此可得，G的任何一课生成树的瓶颈权重都大于或等于5，所以树中包含一条穿过V_3, V_2, V_1, V_4的路径的是一棵最小瓶颈树。但它不是一棵最小生成树，因为这棵树的总权重大于一棵以V_1为根节点，V_2, V_3, V_4为三个叶结点的树

  (b) True. 假设T是G中的一个最小生成树，T’是一棵有更小瓶颈边的生成树。那么T就有一个比T‘所有边的权重都大的边e。所以把e添加到T’中会形成一个cycle C。根据切割性质，e$\notin$任何最小生成树。得证

- 4-29 Given a list of n natural numbers $d_1, d_2, ..., d_n$, show how to decide in polynomial time whether there exists an undirected graph G = (V , E) whose node degrees are precisely the numbers $d_1, d_2, ..., d_n$. (That is, if V ={$v_1, v_2, ..., v_n$}, then the degree of vi should be exactly di.) G should not contain multiple edges between the same pair of nodes, or “loop” edges with both endpoints equal to the same node.

  Solution

  If any $d_i = 0$, It is an isolated node in the graph; so we can delete d_i and continue on the smaller true by recursion.

  If all $d_i > 0$, relabel them so that $d_1 \geq d_2 \geq ... \geq d_n > 0$

  Now subtract 1 from the first d_n numbers. and drop the last number to create list L

  $\exists$ a graph with degree equal to the list $d_1, ..., d_n$, iff $\exists$ a graph with degrees that form list L.

  Proof.

  If $\exists$ graph with degree sequence L, then we can add an n^{th} node with neighbors equal to nodes $v_1, v_2 ..., v_{d_n}$, thereby obtaining a graph with degree sequence $d_1, ... d_n$

  In this case, it must be shown that there is in fact such a graph where node V_n is joined to precisely the nodes $V_1, V_2, ..., V_{d_n}$. After that we can delete node n and obtain the list L.

  Consider any graph G with degree sequence $d_1, ..., d_n$. We transform G into a graph where V_n is joined to $V_1, V_2, ..., V_{d_n}$. If this properly does not hold already, then $\exists$ i < j so that V_n is joined to V_j but not V_i. Since $d_i \geq d_j$, there must be some V_k not equal to any of V_i, V_j, V_n with the property that (V_i, V_k) in an edge but (V_j, V_k) is not. Replace these two edges by (V_i, V_n) and (V_j, V_k)

  This keeps all degrees the same; and repeating this transformation will convert G into a graph with the desired property.

- 5-1 You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains n numerical values—so there are 2n values total—and you may assume that no two values are the same. You’d like to determine the median of this set of 2n values, which we will define here to be the nth smallest value.
  However, the only way you can access these values is through queries to the databases. In a single query, you can specify a value k to one of the two databases, and the chosen database will return the kth smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.
  Give an algorithm that finds the median value using at most O(log n) queries.

  Solution

  运用递归

  median (n, a, b)

  k = $\lceil \frac{1}{2} \rceil$

  if (n == 1) then return min(A(a + k) , B(b + k))

  if A(a + k) < B (b + k)

  then return median (k, a + $\lfloor \frac{1}{2} n \rfloor$, b)

  else return median (k, a , b + $\lfloor \frac{1}{2} n \rfloor$)

- 5-4

Solution

定义两个vector，a = $(q_1, q_2, ..., q_n)$ 和 b = $(n^{-2}, (n-1)^{-2}, ..., \frac{1}{4}, 1, 0, -1, -\frac{1}{4}, ... -n^{-2})$

对于每一个j，a和b的卷积会包含一组 $\sum_{i<j} \frac{q_i}{(j-i)^2} + \sum_{i>j} \frac{-q_i}{(j-i)^2}$, 由此我们乘以$C_{q_j}$得到$F_j$

卷积的时间复杂度为 $O(nlogn)$， 重构F_j则需要额外的时间复杂度O(n)

- 5-7

  Solution

  First, we start at an arbitrary grid, if one of its adjacent grid has a smaller label, then we move to this grid, and repeat the same process. Finally we will reach a local minimum grid.

  Now let's consider a subgraph of the G, suppose it is an rectangular grid graph, and v is the smallest grid of the grids on the border of this rectangle. Suppose V' is the grid adjacent to v and inside the rectangle, if V' is smaller than v, then we can conclude that there exist a local minimum inside the triangle. This is because if we can always a "decreasing path" starting from v', ending at at local minimum. This decreasing path will not go across the border of the rectangle, since v' is smaller than all the grids on the border. So, base on this conclusion, we can divide and conquer the problem.

  We may assume that the outer of the G has a very large label.

  At the first step we probe all the grids of the column at the very middle of G. The column divides the graph G into two n*(n/2) rectangular subgraphs. Suppose the smallest grid of the column is v_1, then we probe the two neighbors of v_1 which are not in the same column of v_1. If the two grids are all larger than v_1, then v_1 is a local minimum. Otherwise, at least one of the two neighbors is smaller than v_1, so we can determine which rectangular subgraph definitely contains a local minimum, according the conclusion we just worked out. And at the second step, we probe all the grids of the row at the very middle of the rectangular subgraph, this row divides the subgraph into two (n/2)*(n/2) square subgraphs. Suppose the smallest grid of the row is v_2, if v_2>v_1, then the decreasing path starting at v_1 will not go across the row, so we choose the square subgraph which contains v_1. (At this situation v_1 will not locate at the common corner of the two squares, since v_1's neighbor is smaller than it and v_2 is larger than it.) If v_2<v_1, then we probe the two neighbor's ofv_2 which are not at the same row of v_2, and use the same way as we due with v_1, to select one square subgraph, or assert v_2 itself is a local minimum. (At this situation v_2 will not locate at the common corner of the two squares, since v_1 is the smallest grid of the column and v_2< v_1.) Now we have shrunk the graph in to a (n/2)*(n/2) subgraph, with 3n/2+4 probes at most. We can apply the same process to the subgraph and finally we will get a local minimum.

  So the total number of probes will be $T(n) = T(n/2) + 3n/2 + 4 = 3n + 4log(n) = O(n)$.