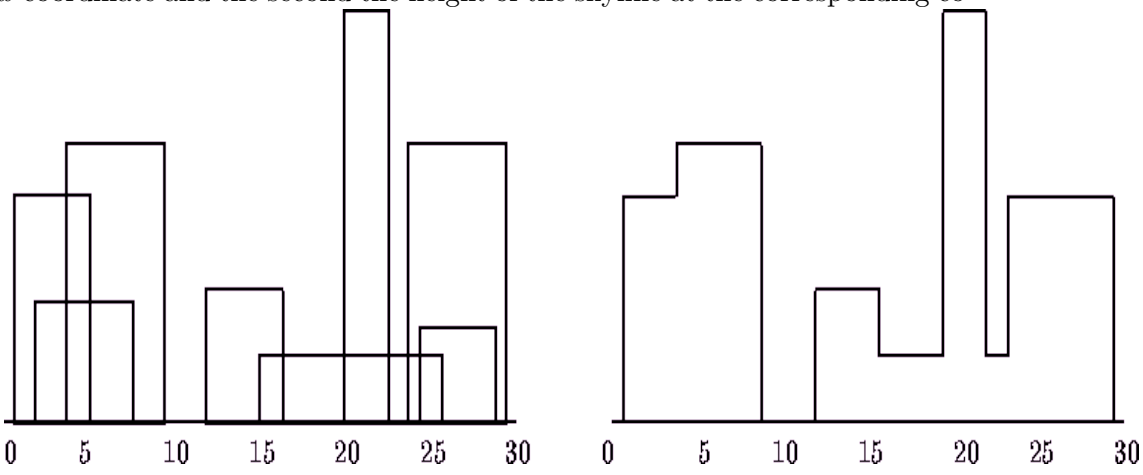
(e) Implement the recurrence efficiently in pseudocode and analyse its time and space complexity. **Solution**

```
M[1] = 0
for i= 2 to n do
  A = M[i-1] + f[i] - f[i-1] , B =  M[p[i]] + s[i] - f[p[i]]
  if A > B then M[i] = B else M[i]= A.
return M[n]
```

(f) How would you modify your algorithm to also produce a list of the jobs scheduled? **Solution** Keep track of which choice is made in the recursion:

```
M[1] = 0, b[1] = 1
for i= 2 to n do
  A = M[i-1] + f[i] - f[i-1] , B =  M[p[i]] + s[i] - f[p[i]]
  if A > B then M[i] = B , b[i] = 0 else M[i]= A. b[i] = 1
i= n
while i > n do if b[i] = 1 then output i, i = p[i]
                           else i = i -1.
```

Problem 4 Manhattan Skyline [10] With the advent of high speed graphics workstations, CAD (computer-aided design) and other areas (CAM, VLSI design) have made increasingly effective use of computers. One of the problems with drawing images is the elimination of hidden lines – lines obscured by other parts of a drawing.
You are to design a program to assist an architect in drawing the skyline of Manhattan given the locations of the buildings in the city. As we know, in Manhattan, all buildings are rectangular in shape and they share a common bottom (the city is very flat). The city is also viewed as two-dimensional. There are n buildings specified by three arrays $L[1 \ldots n], R[1 \ldots n]$ and $H[1 \ldots n]$. Building i has left side at coordinate $L[i]$, right coordinate at $R[i]$ and height $H[i]$. In the figure below, buildings are shown on the left with $L = (12, 2, 14, 19, 3, 1, 24, 23)$ $R = (16, 7, 25, 22, 9, 5, 28, 29)$ and $H = (7, 6, 3, 18, 13, 11, 4, 13)$. The skyline, shown on the right, is represented by the two arrays: $(1, 3, 9, 12, 16, 19, 22, 23, 29)$ and $(11, 13, 0, 7, 3, 18, 3, 13, 0)$ where the first array gives the x coordinate and the second the height of the skyline at the corresponding ordinate.



(a) Give the overall strategy of a Divide-and-Conquer algorithm. **Solution**: Divide the set of buildings into two roughly equal size parts. Compute the skylines of the two subsets recursively, and then combine them into an overall skyline.

(b) Explain clearly what the recursive subproblems should return. **Solution**: Each of the two sub-problems should return two lists representing the skyline of the corresponding skyline: the left subproblem returns the arrays x_L, h_L and thr right subproblem x_R, h_R.

(c) Explain clearly the Conquer step: how will you combine the solutions to the subproblem to get a solution for the original problem? This is the most crucial part! **Solution**: Like Mergesort, we scan the x–arrays from left to right and at each coordinate we find which of the two skyline is higher using the height arrays. Thus we merge the two x arrays and create the corresponding overall h array as we go along.

(d) Write a recurrence for the running time and give a short justification. Deduce the running time of the algorithm in $O()$ notation. **Solution**: The recurrence for the running time is $T(n) = 2T(n/2) + O(n)$ because the divide and combine steps can be done in time $O(n)$. The soltuion to the recurrence is $T(n) = O(n \log n)$.

Problem 5 Taxi Göteborg [10] You are working for Taxi G̈teborg and are dveloping software for an automated response system. At a given point in time, there are m taxis at different locations in the city and there are n customers requesting a taxi, also from various parts of the city. Customer i needs the taxi within t_i minutes. Using some other nifty software using Dijkstra with GPS, you can determine if taxi j can reach customer i within t_i minutes in G̈teborg traffic. Now you need to determine the maximum number of customers you can satisfy. Give an efficient algorithm and analyse its running time. **Solution** First create abipartite graph with taxis on the left, customers on the right, and connect taxi j to customer i if your GPS software says it can raech the customer in time t_i. Now create aflow network with a source s having edges to all taxis of capacity 1 a sink t which has edges from all customers with capacity 1 and edges int he bipartitte graph are of capacity 1 and directed from taxis to customers. There is a flow of integer value f from s to t exactly if f customers can be satisfied hence the maximum number of satisfied customers equals the max flow in the network. Since all capacities are 1, the Ford Fulkerson algorithm can compute this in time $O(nm)$.

Problem 6 Multiple Interval Scheduling [10] We've seen the Interval Scheduling problem in class; here we consider a computationally much harder version of it that we'll call Multiple Interval Scheduling.As before, you have a processor that is available to run jobs over some period of time. (E.g. 9 AM to 5 PM.)
People submit jobs to run on the processor; the processor can only work on one job at any single point in time. Jobs in this model, however, are more complicated than we've seen in the past: each job requires a *set* of intervals of time during which it needs to use the processor. Thus, for example, a single job could require the processor from 10 AM to 11 AM, and again from 2 PM to 3 PM. If you accept this job, it ties up your processor during those two hours, but you could still accept jobs that need any other time periods (including the hours from 11 to 2).
Now, you're given a set of n jobs, each specified by a set of time intervals, and you want to schedule as many jobs as possible. Call this problem OPT-MS. Consider the