A6:Solutions

Tuesday, May 4, 2021 6:30 PM

1) Knapsack with duplicates

Item i has size s_i & value v_i. Total capacity of knapsack is C & multiple copies of an item are allowed.

Order items arbitrarily

Define subproblems as follows

$X[i,j]$ = maximum value of items picked from set $\{1..i\}$ (with duplicates) of total size j.

Consider the (multi)set of items S which give value $X[i,j]$.

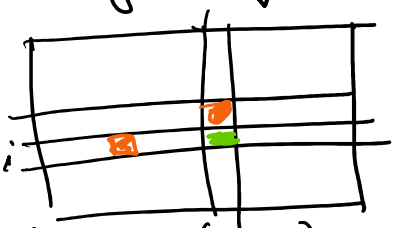
1) $i \in S$: Then $X[i,j] = X[i,j-s_i] + v_i$

2) $i \notin S$: Then $X[i,j] = X[i-1,j]$

Thus $X[i,j] = \max\left(X[i-1,j], X[i,j-s_i] + v_i \right)$. Note $X[i,0] = X[0,j] = 0$

Consider a matrix X with n rows & C columns. The $(i,j)^{th}$ entry of this matrix corresponds to $X[i,j]$.

Note that the matrix can be filled row by row from left to right since to compute an entry we need entries of the previous row or those in the same row to the left. Since time reqd. to compute each entry is constant, the table can be filled in $O(nC)$ time.



The solution to the problem is the maximum entry in the last row, say this is $X[n,f]$. To determine the multiset of items which give this value we trace how the value is obtained by traversing the table. Every time we move left within a row we include one copy of the item corresponding to this row.

The time required is $O(n+C)$ since in each step we either reduce the row number or the column number.

Thus total time required is $O(nC)$

Box stacking

Box i has dimensions h_i, w_i, d_i. A box i can be stacked on top of box j if its base has strictly smaller dimensions than that of box j. Boxes can be rotated too.

Construct a graph G whose vertices correspond to the choice of base for the boxes. Thus for box i we have 3 vertices corresponding to base (h_i, w_i), (h_i, d_i) & (w_i, d_i). Each vertex has a value corresponding to the height for that particular rotation of the box; Thus vertex (h_i, w_i) has value d_i.

G has an edge from (x_i, y_i) to (x_j, y_j) if $(x_j < x_i)$ and $(y_j < y_i)$ or $(y_j < x_i)$ and $(x_j < y_i)$. Thus this edge captures the fact that box j (with base (x_j, y_j)) can be stacked on top of box i with base (x_i, y_i). When we follow an edge the area of the base decreases & hence G cannot have a cycle.

Thus G is a DAG. A path in G corresponds to a stacking & the total value of vertices on the path is the height of the stacking. Thus we have to find the longest path in a DAG where the length of a path is the sum of the values of vertices along the path.

Consider a topological sort of G. Let v_i be the ith vertex in this ordering & d_i be its value.

We next define the subproblems. Let $X[i]$ be the length of the longest path ending at v_i. If the vertex preceding v_i on the longest path ending at v_i is v_j then $(v_j, v_i) \in E$ and $X[i] = X[j] + d_i$. Thus

$$X[i] = \max_{(v_j, v_i) \in E} X[j] + d_i$$

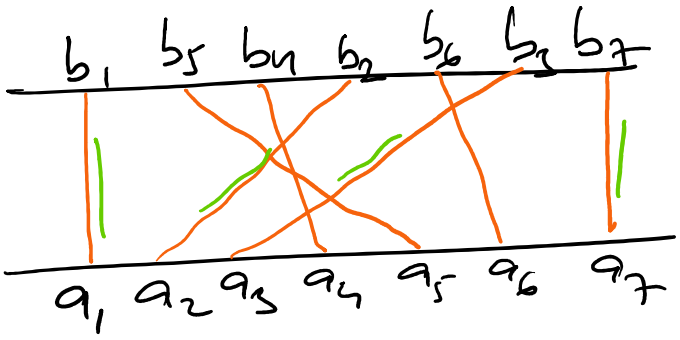
Note that if v_i has no incoming edge then $X[i] = d_i$.

We consider vertices in the topological ordering & compute each $X[i]$ can be computed in

We consider vertices in the topological ordering of G and compute X[i] for each v_i. Since each X[i] can be computed in time proportional to the no. of incoming edges to v_i, the total time used to compute X[] is O(m) where m is the number of edges in G. Since G has 3n vertices, m is $O(n^2)$.

The height of the best stacking is given by the largest value of X[]; suppose this is X[k]. We can determine the longest path in G corresponding to X[k] by tracing how the value X[k] was obtained in our procedure. The path then gives a stacking of the boxes.

Building Bridges



For simplicity assume all coordinates b_i & b_j are distinct. We also assume the numbering is such that

$a_1 < a_2 < a_3 < \ldots < a_n$

Construct an array c, where c[i] = j where b_j is the i^{th} smallest coordinate in $\{b_1 \ldots b_n\}$. For the above example

$c = [1, 5, 4, 2, 6, 3, 7]$.

Let $c[i_1] < c[i_2] < \ldots < c[i_k]$ be an increasing subsequence

Then $a_{c[i_1]} < a_{c[i_2]} < \ldots < a_{c[i_k]}$ & by construction

$b_{c[i_1]} < b_{c[i_2]} < \ldots < b_{c[i_k]}$. Thus $c[i_1], c[i_2] \ldots c[i_k]$ are indexes of cities between which non-crossing bridges can be built.

Conversely the indexes of cities between which non-crossing bridges can be built defines an increasing subsequence in c. Thus finding the largest such collection of bridges corresponds to

bridges ... bridges ...
Thus finding the largest such collection of bridges corresponds to finding the longest increasing subsequence in c which can be done in $O(n^2)$ time [as done in class].

Balanced partitions

We wish to partition a set of n numbers in $[1..k]$ into 2 subsets whose difference of sums is minimised. S_1, S_2 are the sums of the two subsets. Let $S_1 \geq S_2$ & $S_1 + S_2 = W$. Then $|S_1 - S_2| = S_1 - S_2$ $= W - 2S_2 = 2(\frac{W}{2} - S_2)$. Thus minimizing $|S_1 - S_2|$ is identical to finding a set X whose sum of elements is less than $W/2$ & as close to $W/2$ as possible.

This can be solved by following the procedure for the subset sum problem solved in class. We order the n numbers arbitrarily & let $X[i,j] = 1$ if there exists a subset of the first i numbers which sums to j

0 o.w.

Consider a matrix X with n rows & $\lfloor W/2 \rfloor$ columns whose $(i,j)^{th}$ entry corresponds to $X[i,j]$. This is identical to the matrix created for the subset sum problem & can be filled using the recurrence

$$X[i,j] = X[i-1,j] \text{ OR } X[i, j-a_i]$$

where a_i is the i^{th} number in the ordering.

Once we have computed all entries of this matrix X we consider the rightmost 1 in the last row; let this be $X[n,k]$. This implies there exists a subset whose elements sum to k; the set itself can be determined by tracing how the value $X[n,k] = 1$ was obtained. Let S be the set so obtained. Then the partition of the n numbers is the set S & its complement & the difference in their sums is $W - k - k$ $= W - 2k = 2(\frac{W}{2} - k)$.

This is the best partition possible. If there were a ... with smaller difference then the smaller set

This is the best partition possible. If there is a different partition with smaller difference then the smaller set in this partition would have a sum less than $W/2$ but larger than k. This is a contradiction to our choice of set S.

The running time for the subset sum procedure is $O(nW)$ since $W \leq n \cdot K$ the time is $O(n^2 K)$.