Suggested Solutions to HW #4

3. (5.11) Suppose that there are two different (maybe proposed) skylines: One is projected on a screen with a blue color, and the other is superimposed on the first one with a red color. Design an efficient algorithm to compute the shape that will be colored purple. In other words, compute the intersection of two skylines. Please present your algorithm in suitable pseudo code.

Solution.(Jen-Feng Shih)

Algorithm Intersection_of_Skyline;

input: b (an array of size m)
r (an array of size n)
output: p (an array of size q)
$i, j, k := 1$; // The index of array b, r, p
if $(b[i] < r[j])$ **then**
$\quad$ **while** $(b[i+2] < r[j])$ **do**
$\quad$ $i := i + 2$;
else
$\quad$ **while** $(r[j+2] < b[i])$ **do**
$\quad$ $j := j + 2$;
// Find the first location of the intersection
while $(min(b[i+1], r[j+1]) = 0)$ **do**
$\quad$ **if** $(b[i] < r[j])$ **then** $i := i + 2$;
$\quad$ **else if** $(b[i] > r[j])$ **then** $j := j + 2$;
$\quad$ **else if** $(b[i+2] < r[j+2])$ **then** $i := i + 2$;
$\quad$ **else if** $(b[i+2] > r[j+2])$ **then** $j := j + 2$;
$\quad$ **else** $i := i + 2$; $j := j + 2$;
$p[k] := max(b[i], r[j])$;
// Avoid the case where the first height is 0, for example (1, **0**, 3, 5, 7) = (3, 5, 7).
while $(i < m$ && $j < n)$ **do**
$\quad$ **if** $(p[k-1] \neq min(b[i+1], r[j+1]))$ **then**
$\quad\quad$ $p[k+1] := min(b[i+1], r[j+1])$;
$\quad\quad$ $k := k + 2$;
// Avoid the index with the same height as before for example(1, **5**, 6, **5**, 8) = (1, 5, 8).
$\quad$ **if** $(b[i+2] < r[j+2])$ **then**
$\quad\quad$ $p[k] := b[i+2]$;
$\quad\quad$ $i := i + 2$;

else if $(b[i+2] > r[j+2])$ **then**
$p[k] := r[j+2];$
$j := j+2;$
else
$p[k] := r[j+2];$
$i := i+2;$
$j := j+2;$

□

4. (5.17) The Knapsack Problem that we discussed in class is defined as follows: Given a set S of n items, where the ith item has an integer size $S[i]$, and an integer K, find a subset of the items whose sizes sum to exactly K or determine that no such subset exists.

We have described in class an algorithm to solve the problem. Modify the algorithm to solve a variation of the knapsack problem where each item has an *unlimited* supply. In your algorithm, please change the type of $P[i,k].belong$ into integer and use it to record the number of copies of item i needed.

Solution.

Algorithm Knapsack (S, K)**;**

begin
$P[0,0].exist := true;$
$P[0,0].belong := 0;$
for $k := 1$ **to** K **do**
$P[0,k].exist := false;$
for $i := 1$ **to** n **do**
for $k := 0$ **to** K **do**
$P[i,k].exist := false;$
if $P[i-1,k].exist$ **then**
$P[i,k].exist := true;$
$P[i,k].belong := 0;$
else if $k - S[i] \geq 0$ **then**
if $P[i,k-S[i]].exist$ **then**
$P[i,k].exist := true;$
$P[i,k].belong := P[i,k-S[i]].belong + 1;$
end

□