Q 1 in 12.2

1). For a minimization problem, a min-heap would be used, and for a maximization problem, a max-heap would be used.

Q 6. in 12.2

(a) Consider the items sorted in nonincreasing order of their payoff values per unit weight $v_1/w_1 \geq \dots > v_n/w_n$.

A node in branch-&-bound tree represents a subset S of the k items $S = \{ item\ i_1, \dots, item\ i_k \}$ with total weight $w(S)$ and total value $v(S)$, respectively.

A more sophisticated upper bound $UB(S)$ on the value of any subset that can be obtained by adding items to S can be computed as follows. The consecutive values of the items not included in S would be added to $v(S)$ as long as the total weight doesn't exceed the knapsack capacity. When encountering an item that violates this constraint, its fraction needed to fill the remaining capacity is multiplied by the item's value, and this product is added to previously added in computing $UB(S)$

Ex.

For items

	1	2	3	4
w_i	4	7	5	3
v_i	40	42	25	12
v_i/w_i	10	6	5	4

$W = 10$

For root node,

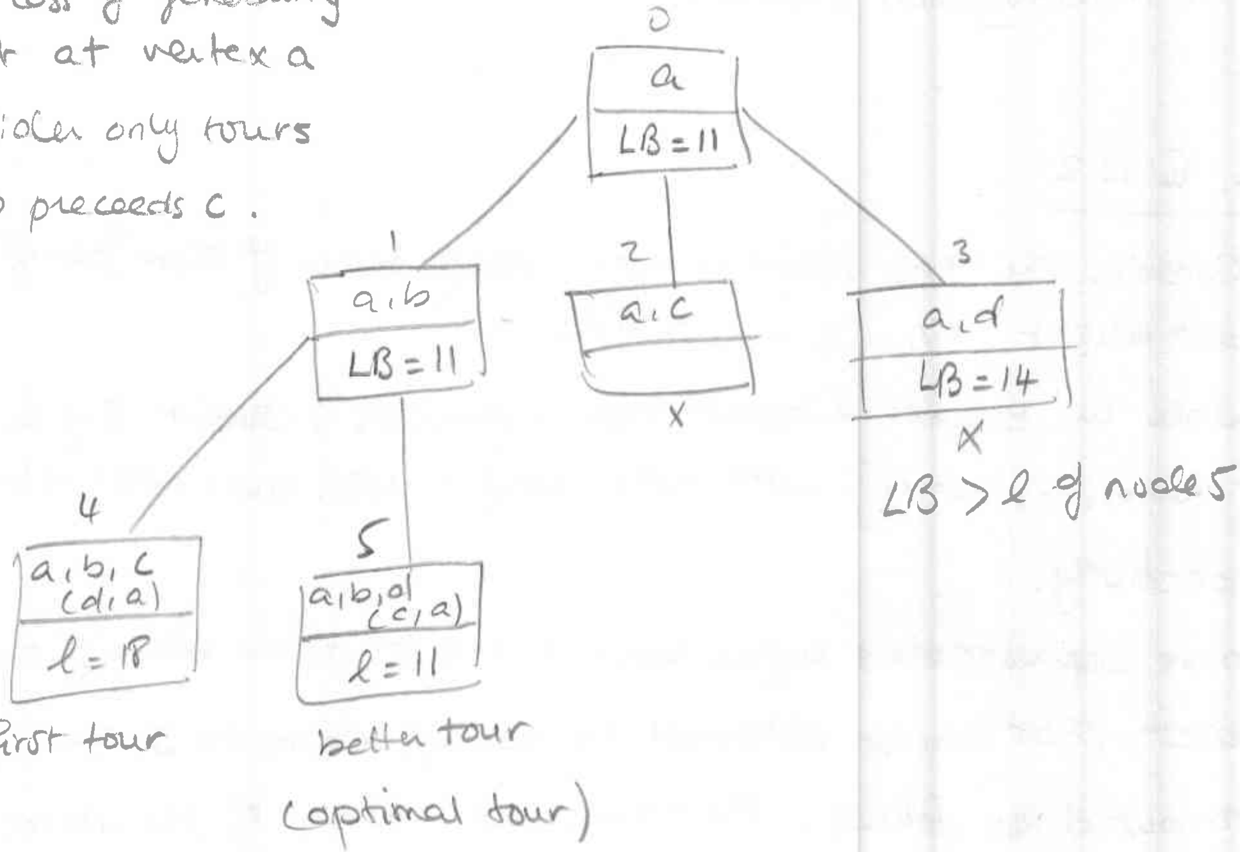
$UB(S) = 40 + \frac{6}{7} * 42$

$= 76$

Q 8. (a) in 12.2. See Lecture.

Q 9. in 12.2

without loss of generality
- start at vertex a
- consider only tours where b precedes c.



Optimal tour is a, b, d, c, a of length 11.

Q 2. Ex. 12.3

(a) Alg. NearestNeighborTSP (D[1..n, 1..n] , s)

// Nearest Neighbor Heuristic alg for TSP, taking as i/p
// intercity dist matrix D and starting vertex s. O/P Tour.

for i = 1 to n do
    Visited [i] = FALSE
Initialize a list Tour with s
Visited [s] = TRUE
current = s
for i = 2 to n do
    - find column j with smallest element in row current AND
      Visited [j] = FALSE
    - current = j
    - Visited [j] = TRUE
    - add j to end of list Tour
endfor
add s to end of Tour
return Tour

(b) The time efficiency is $\Theta(n^2)$ since finding column j inside the i-loop is $\Theta(n)$.

Q3. in 12.3

Making the walk in clockwise direction, we get the tour

a, b, d, e, c, a of length 38. The length is not the

same as that of the tour in Fig 12.11b.

Q 5. in 12.3

Step 1 takes $\Theta(n)$

Step 2 can be done in $\Theta(n \log n)$

Step 3 takes $\Theta(n)$

$\Rightarrow$ Overall $\Theta(n \log n)$
