SEARCH TREES: BRANCH AND BOUND

CS340

Branch and Bound

- Used for optimization problems
  - Trying to find the optimal solution (as opposed to feasible solutions)
- Requires 2 more pieces of data
  - A way to provide a bound on the best value that can be obtained by continuing with the current partial solution (= a way to rank partial solutions)
  - The value of the best solution seen so far
- Best-first
  - Instead of trying options in order, try the most promising ones first

Assignment Problem

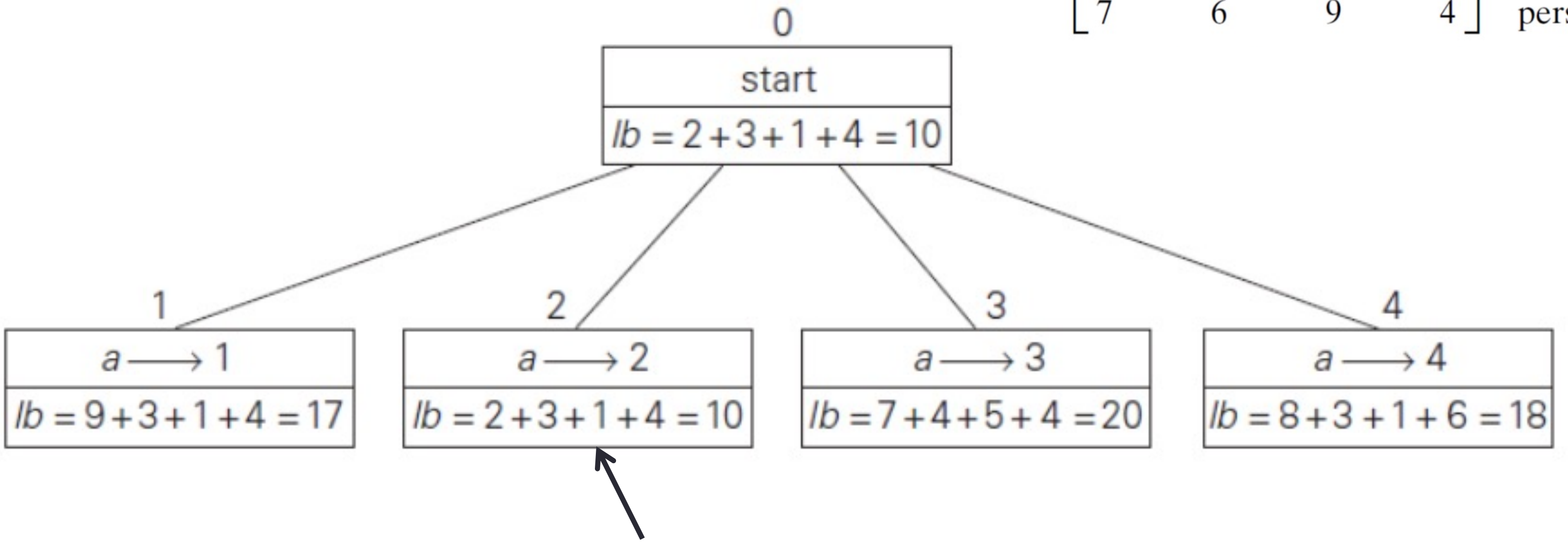
- Assign n jobs to n people such that the total cost is as low as possible
- What is a lower bound on the cost?

$$
\begin{array}{cc}
 & \begin{array}{cccc} \text{job 1} & \text{job 2} & \text{job 3} & \text{job 4} \end{array} \\
C = & \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix}
\end{array}
\begin{array}{l} \\ \text{person } a \\ \text{person } b \\ \text{person } c \\ \text{person } d \end{array}
$$

Assignment Problem

$$C = \begin{array}{c} \\ \\ \\ \\ \\ \end{array}\begin{array}{cccc} \text{job 1} & \text{job 2} & \text{job 3} & \text{job 4} \end{array}$$

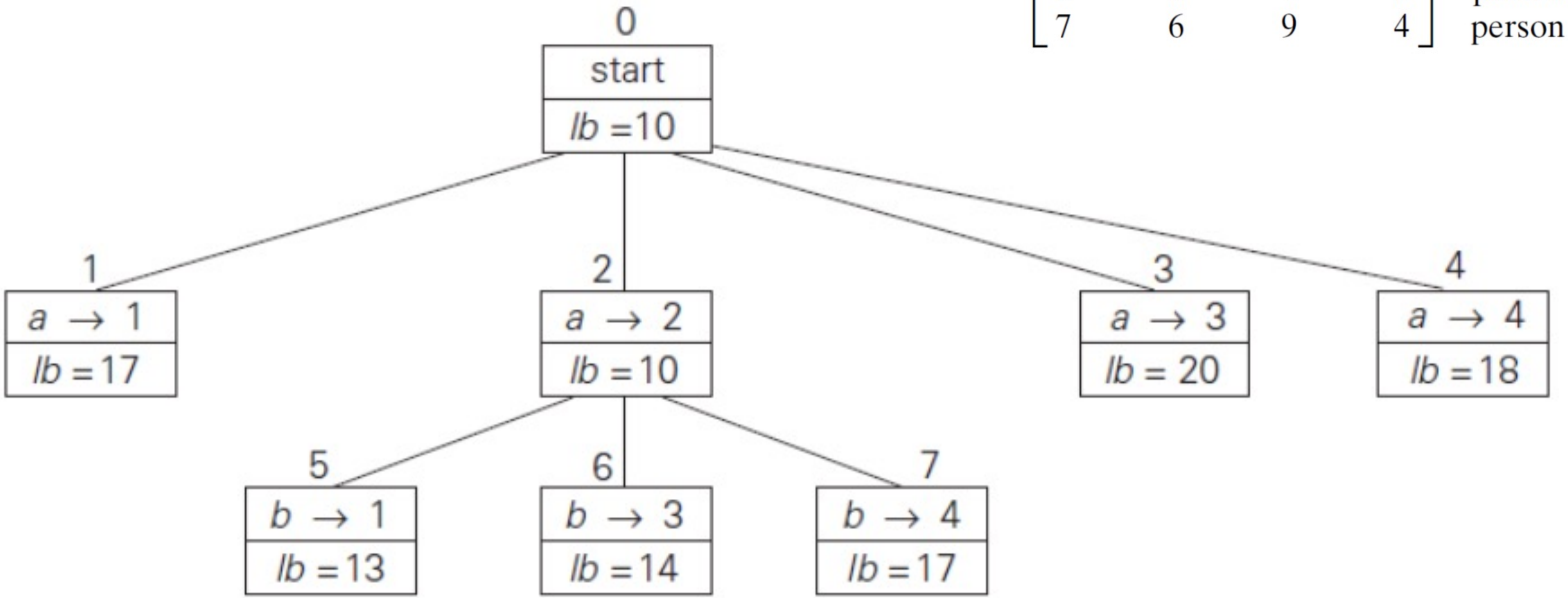
$$C = \left[\begin{array}{cccc} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{array}\right]\begin{array}{l} \text{person } a \\ \text{person } b \\ \text{person } c \\ \text{person } d \end{array}$$



Most promising

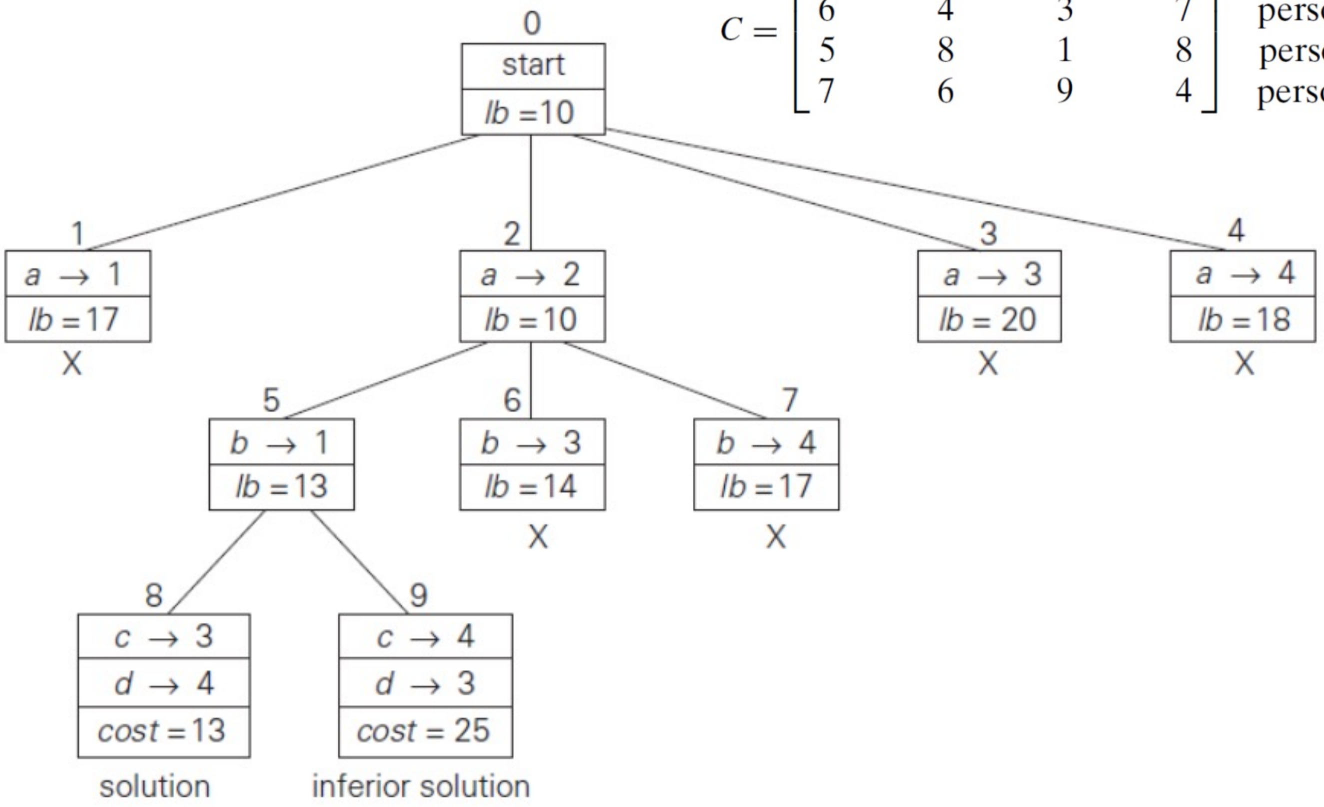
Assignment Problem

$$C = \begin{array}{c} \begin{matrix} \text{job 1} & \text{job 2} & \text{job 3} & \text{job 4} \end{matrix} \\ \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix} \end{array} \begin{matrix} \\ \text{person } a \\ \text{person } b \\ \text{person } c \\ \text{person } d \end{matrix}$$



Assignment Problem

$$
\begin{array}{cccccc}
 & \text{job 1} & \text{job 2} & \text{job 3} & \text{job 4} & \\
C = & \left[\begin{matrix} 9 \\ 6 \\ 5 \\ 7 \end{matrix}\right. & \begin{matrix} 2 \\ 4 \\ 8 \\ 6 \end{matrix} & \begin{matrix} 7 \\ 3 \\ 1 \\ 9 \end{matrix} & \left.\begin{matrix} 8 \\ 7 \\ 8 \\ 4 \end{matrix}\right] & \begin{matrix} \text{person } a \\ \text{person } b \\ \text{person } c \\ \text{person } d \end{matrix}
\end{array}
$$



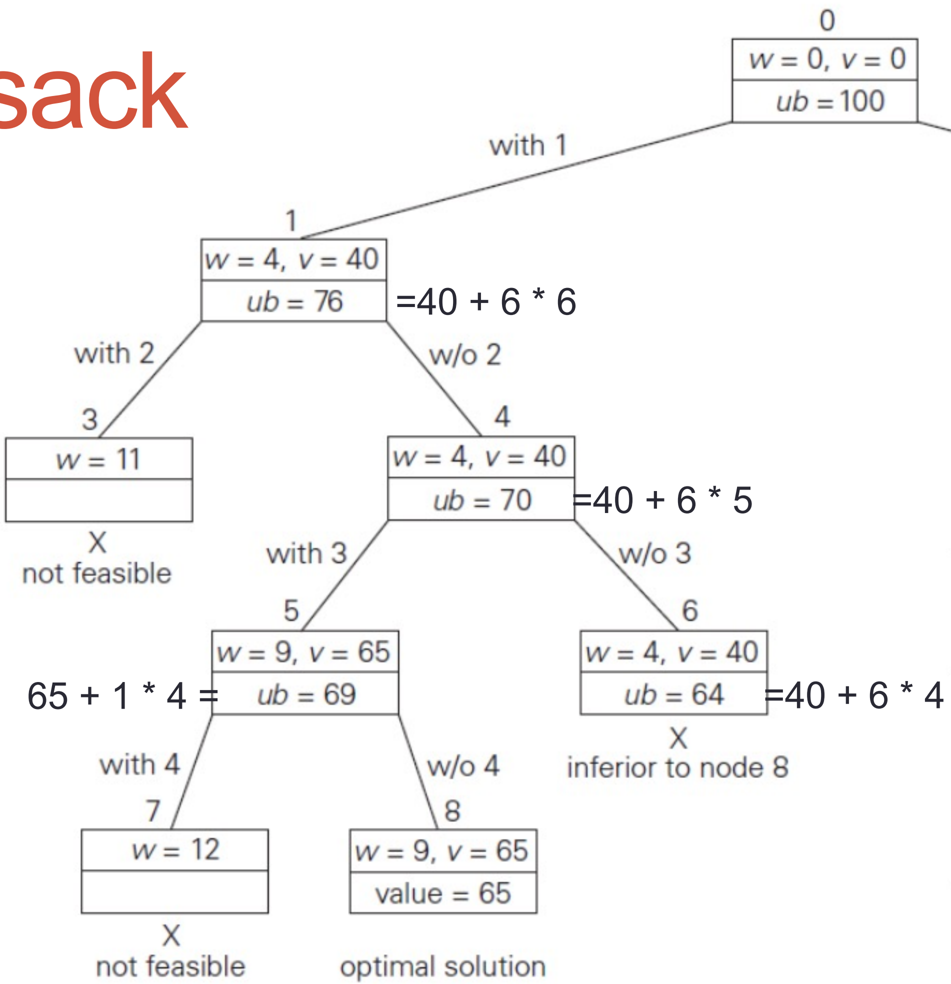
Knapsack

- Order items by weight-to-value ratio (descending)
- Upper bound =
  - current items + (remaining capacity * best payoff among remaining items)
- Every node is a potential solution, therefore, evaluate best solution seen so far at each node, instead of at each leaf

item	weight	value	$\frac{\text{value}}{\text{weight}}$
1	4	\$40	10
2	7	\$42	6
3	5	\$25	5
4	3	\$12	4

Capacity W = 10

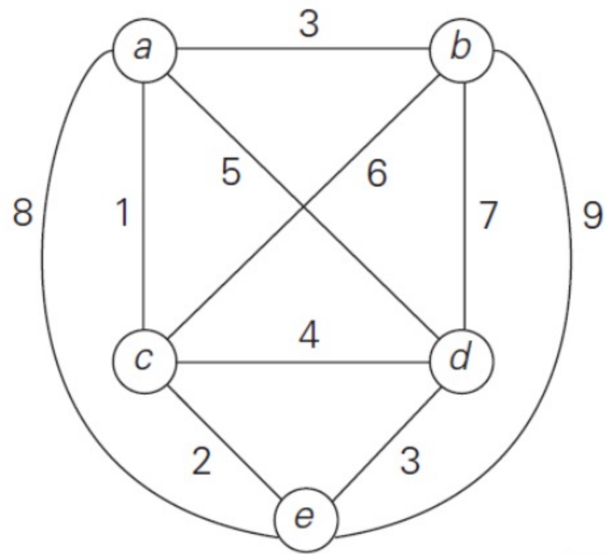
Knapsack



item	weight	value	$\frac{\text{value}}{\text{weight}}$
1	4	\$40	10
2	7	\$42	6
3	5	\$25	5
4	3	\$12	4

Capacity W = 10

Traveling Salesman

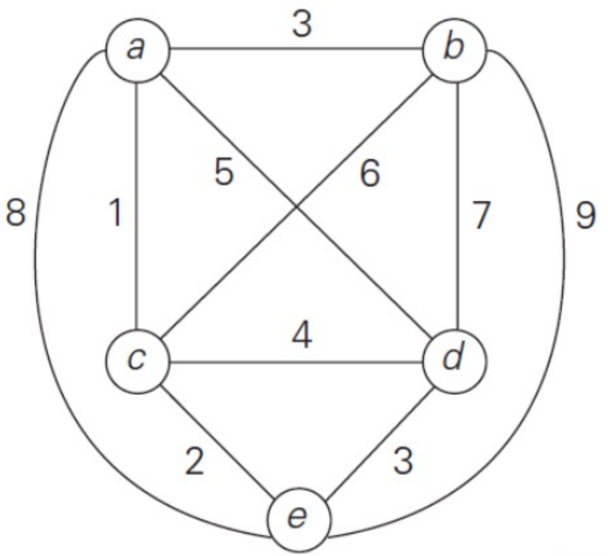


- One way to compute lower bound:
  - shortest distance * |V| = 5 (not so helpful)
- A clever way to compute lower bound:
  - for each city, *i*, find the sum s_i of the distances to the two nearest cities
  - compute the sum, *s*, of these n numbers
  - divide by 2, and round up to nearest integer
  - Why does this work?

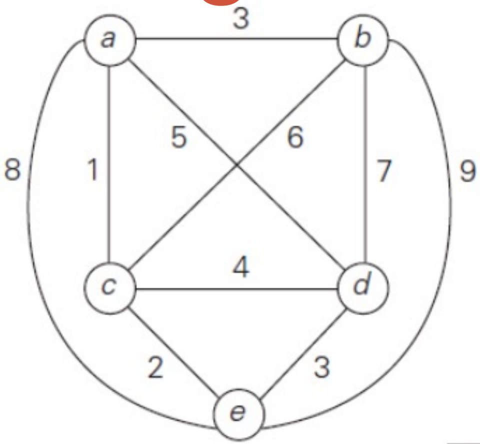
- For this example,
  - lower bound = [(1+ 3) + (3 + 6) + (1+ 2) + (3 + 4) + (2 + 3)]/2 = 14

Traveling Salesman

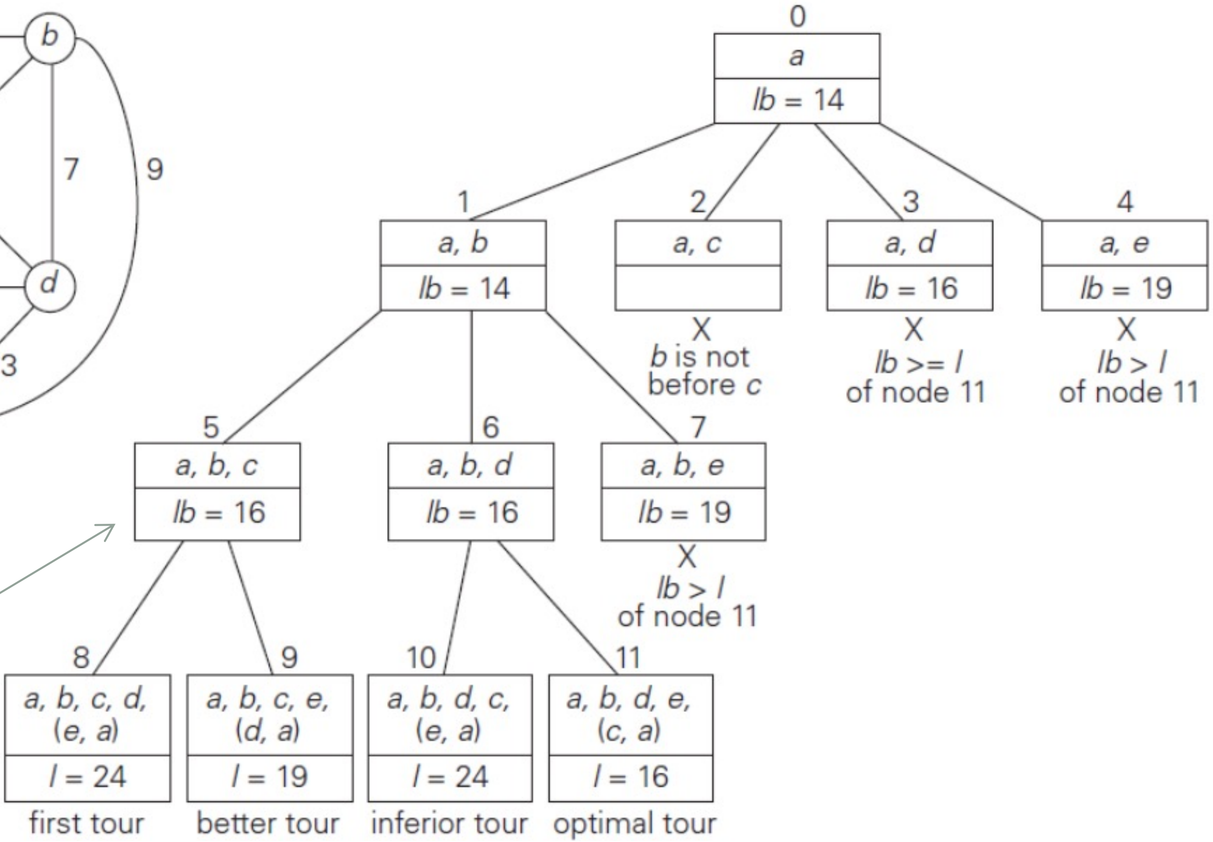
- A few more observations
  - A tour starts where it ends, therefore any starting point will work
    - We only need to consider tours starting at vertex A
  - The direction of the tour doesn't matter.
    - We only need to consider tours going in one direction
    - We will only consider tours where b is visited before c
  - After visiting n-1 cities, the only choice is to visit the last and return home



Traveling Salesman



(a)



Init LB = [(1+ 3) + (3 + 6) + (1+ 2) + (3 + 4) + (2 + 3)]/2 = 14

ABC LB = [(1+ 3) + (3 + 6) + (6 + 1) + (3 + 4) + (2 + 3)]/2 = 16

ABD LB = [(1+ 3) + (3 + 7) + (1 + 2) + (3 + 7) + (2 + 3)]/2 = 16

ABE LB = [(1+ 3) + (3 + 9) + (1 + 2) + (3 + 4) + (2 + 9)]/2 = 19

Branch and Bound

- The hope is that we can do a lot of pruning
- It's impossible to know in advance
- What's the worst case?
  - Why would the worst case be achieved?
- Challenges and opportunities
  - Finding a good bounding function
  - Choosing the order of node generation