Solving the Traveling Salesman Problem (TSP) with Branch-and-Bound

- without loss of generality, we assume that the tour starts at vertex v_1
- example solution

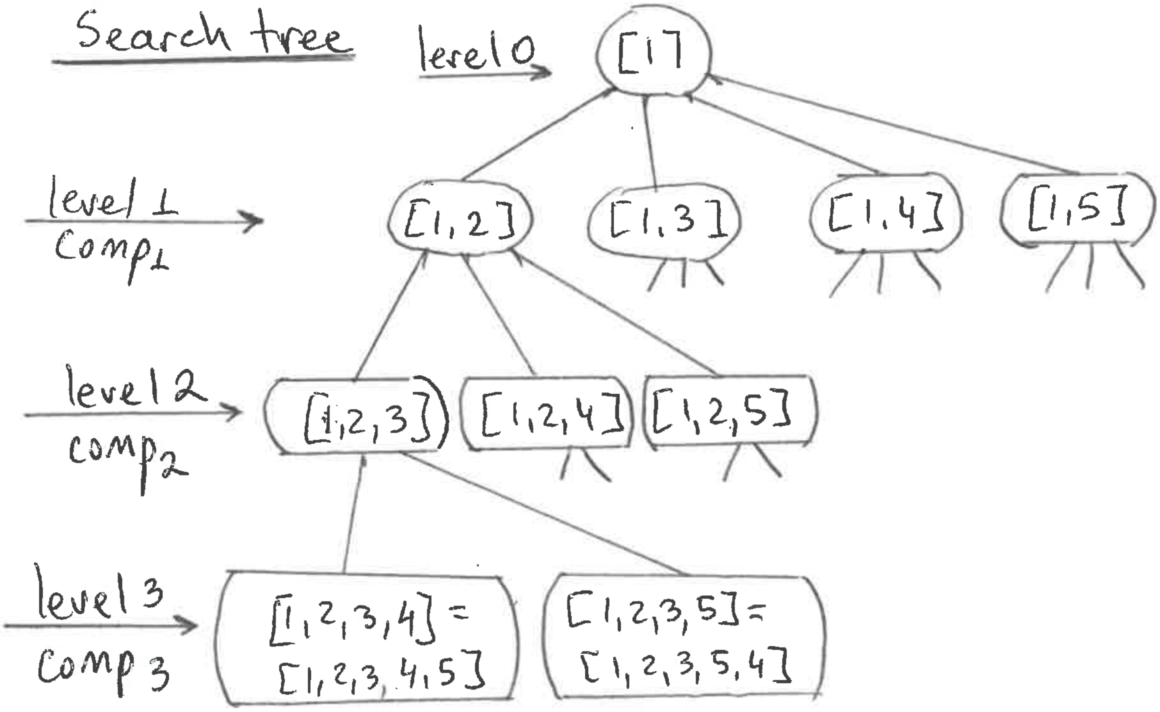
$$\langle \underbrace{v_1}_{\text{start from vertex } v_1}, \underbrace{v_4, v_5, v_2}_{}, \underbrace{v_3}_{\text{only vertex not in the path}} \rangle \qquad n = 5$$

the algorithm only needs to decide the order of $(n-2)$ vertices

- we write the solution as [1, 4, 5, 2, 3]
- formulate the solution using components

$$\text{solution} = (\underbrace{comp_1}_{\text{select second vertex in the tour}}, comp_2, \ldots, \underbrace{comp_{n-2}}_{\text{select the vertex } (n-1) \text{ in the tour}})$$

Search tree



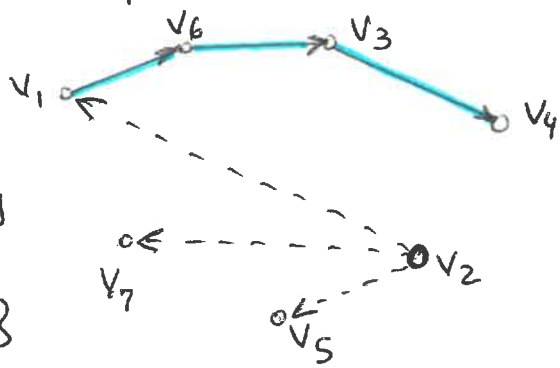
- How do we compute the bound of a node?
  - note that any tour must leave each vertex exactly once
  - a lower-bound is to take minimum edge leaving every vertex

root node, bound $= 4+7+4+2+4 = 21$

This is a lower-bound on the length of a tour

- How do we compute the bound of a partial solution?

[1, 6, 3, 4, ...] $n=7$



v_1: $w(v_1, v_6)$

v_2: $\min\{w(v_2,v_1), w(v_2,v_7), w(v_2,v_5)\}$

v_3: $w(v_3, v_4)$

v_4: $\min\{w(v_4,v_2), w(v_4,v_5), w(v_4,v_7)\}$

v_5: $\min\{w(v_5,v_2), w(v_5,v_7), w(v_5,v_1)\}$

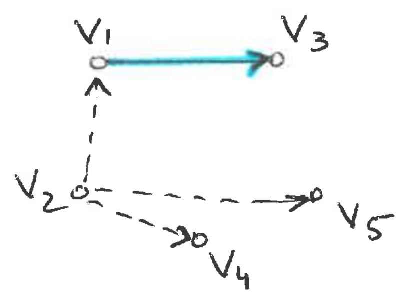
v_6: $w(v_6, v_3)$

v_7: $\min\{w(v_7,v_1), w(v_7,v_5), w(v_7,v_2)\}$

bound = summation of these values

example

[1, 3, –, –, –] $n=5$



v_1: $w(v_1, v_3) = 4$

v_2: $\min\{w(v_2,v_1), w(v_2,v_5), w(v_2,v_4)\} =$

$= \min\{14, 7, 8\} = 7$

v_3: $\min\{w(v_3,v_2), w(v_3,v_4), w(v_3,v_5)\} = \min\{5, 7, 16\} = 5$

v_4: $\min\{w(v_4,v_1), w(v_4,v_2), w(v_4,v_5)\} = \min\{11, 7, 2\} = 2$

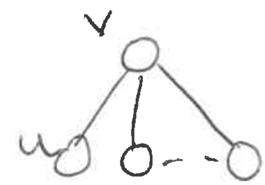
v_5: $\min\{w(v_5,v_1), w(v_5,v_2), w(v_5,v_4)\} = \min\{18, 7, 4\} = 4$

bound $= 4+7+5+2+4 = 22$

Traveling Salesman Problem

TSP-BestFS-Branch-and-bound pruning(n,W[ ][ ],opttour, minlength)

```
PQ = ∅
r.level = 0
r.path = [1]
r.bound = bound(r)          → O(n^2)
minlength = ∞
insert(PQ,r)                → O(lg n)
while PQ ≠ ∅
    v = remove(PQ)          → O(lg n)
    if v.bound < minlength
        u.level = v.level+1
        for all i such that 2 ≤ i ≤ n and i is not in v.path
            u.path = v.path
            add i at the end of u.path
            if u.level == n-2    // check if next vertex completes the tour
                put index of only vertex not in u.path at the end of u.path
                put 1 at the end of u.path
                if length(u) < minlength
                    minlength = length(u)
                    opttour = u.path
            else
                u.bound = bound(u)      → O(n^2)
                if u.bound < minlength
                    insert(PQ,u)        → O(lg n)
```



RT analysis

PQ - priority queue

- insert() and remove() operations take $O(\lg n)$ for an n-element priority queue
- the number of nodes in the search tree is upper bounded by:

$$\leq 1 + (n-1) + (n-1)(n-2) + (n-1)(n-2)(n-3) + \dots + (n-1)(n-2)\cdot\dots\cdot 2$$

- bound() takes $O(n^2)$

$$RT \leq n^2\left(1 + (n-1) + (n-1)(n-2) + (n-1)(n-2)(n-3) + \dots + (n-1)(n-2)\cdots 2\right) =$$

$$= n^2 \cdot (n-1)!\left(\underbrace{\frac{1}{(n-1)!} + \frac{1}{(n-2)!} + \frac{1}{(n-3)!} + \dots + \frac{1}{1!}}_{e-1}\right)$$

$$\boxed{\sum_{i=0}^{\infty} \frac{1}{i!} = e}$$

$e = 2.781\dots$

$$\frac{1}{1} + \underbrace{\frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \dots}_{} = e$$

$$RT \leq (e-1)\cdot n \cdot n!$$

$$\boxed{RT = O(n \cdot n!)}$$