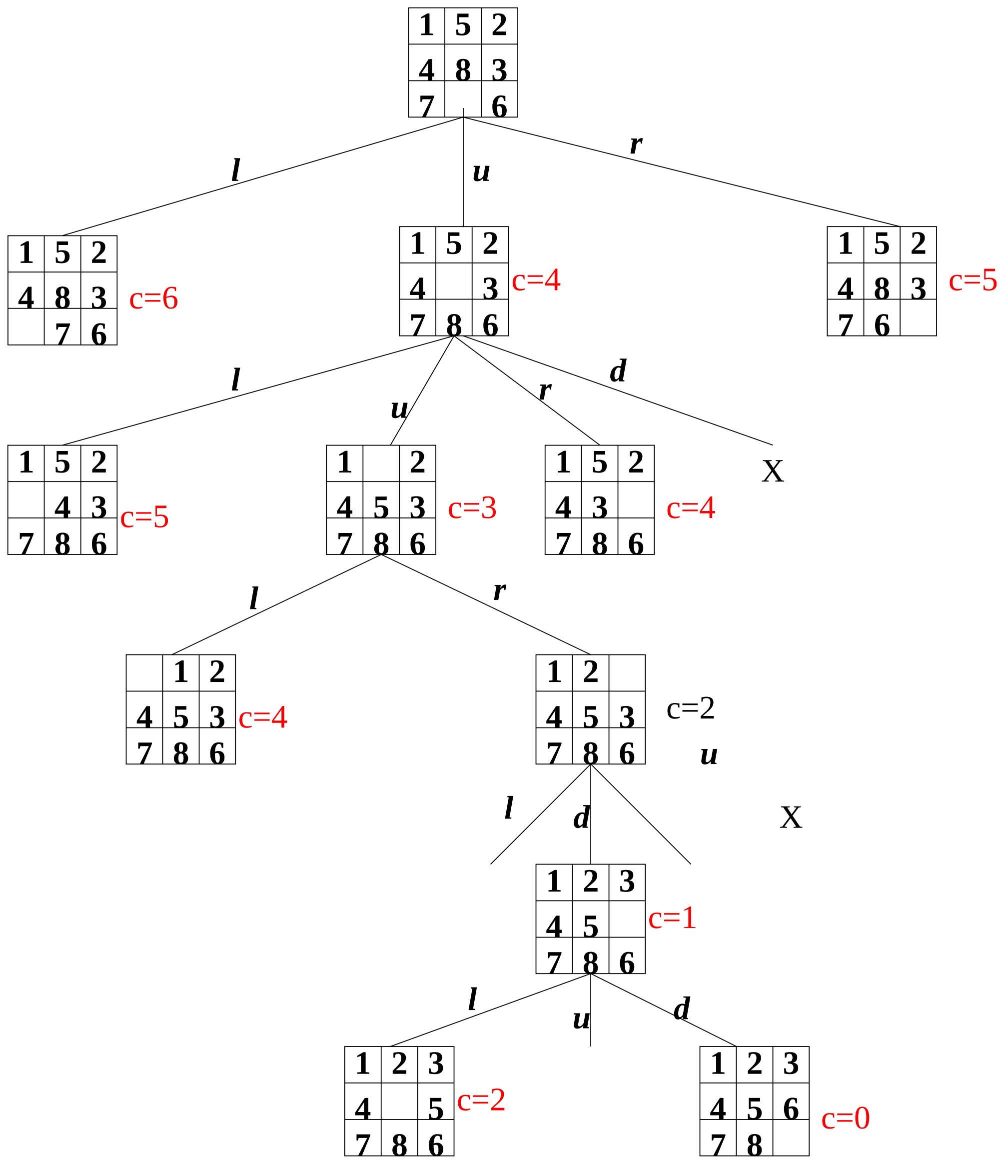
A Puzzle

- When a best-first search is attempted then there is an associated bounding function.
- The bounding function estimates the cost of the exploring a node.
- In case of 8-puzzle the cost function could be the number of tiles out of place, or the Manhattan distance of the nodes out of place.

  Manhattan distance is equal to the sum of the differences in the coordinates of the tile at its current position and at the goal position respectively.

- The empty slot is not considered as a tile.
- With the best-first search with the above mentioned cost function for the 8-puzzle is shown next.

A Puzzle



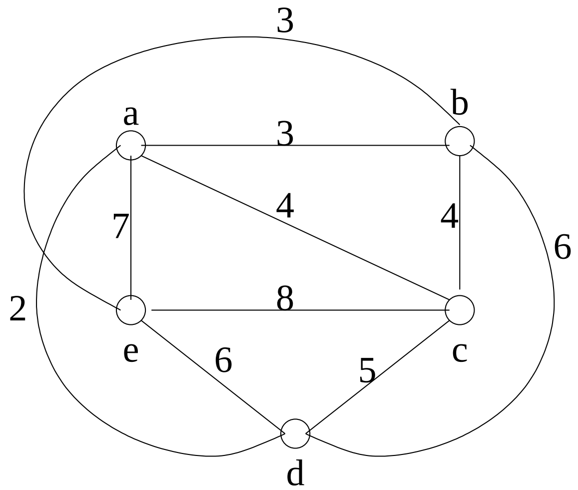
1 5 2
4 8 3
7 6
l
u
r
1 5 2
4 8 3
7 6
c=6
1 5 2
4 3
7 8 6
c=4
1 5 2
4 8 3
7 6
c=5
l
u
r
d
X
1 5 2
4 3
7 8 6
c=5
1 2
4 5 3
7 8 6
c=3
1 5 2
4 3
7 8 6
c=4
l
r
1 2
4 5 3
7 8 6
c=4
1 2
4 5 3
7 8 6
c=2
l
d
u
X
1 2 3
4 5
7 8 6
c=1
l
u
d
1 2 3
4 5
7 8 6
c=2
1 2 3
4 5 6
7 8
c=0

Desirable Properties of c

- Following are desirable properties.
  - c should approximate the cost to reach the solution nicely.
  - c should be easy to compute.
- Usually c computed for a vertex x has two parts, namely
  - An estimate of the cost to reach the solution vertex from the vertex x.
  - A weighted function of the cost to reach x from the root.
  - In case of 8-puzzle, the second part of the cost we have used is 0 whereas the first part is the number of vertices out of place.

Traveling Salesman

- No tour can cost less than the half of the sum over all nodes of the cost of two lowest cost edges incident on these vertices.
- This is because a Hamiltonian cycle of lowest cost is the solution of TSP.



	2 lowest cost edges	sum
a	ad, ab	5
b	ba, be	6
c	ca, cb	8
d	da, dc	7
e	eb, ed	9

- The sum of two least cost edges incident on each vertex is 35. So lower bound of tour = 17.5.

TSP

- Each node in the implicit search tree for solution would represent either
  - a set of edges included in the tour,
  - or, set of edges not included in the tour.
- Suppose we must include (a, e) and exclude (b, c), then lower bound is revised as follows.
- Which means 4 vertices are affected. So we revise cost lower bounds with respect these vertices only.

TSP

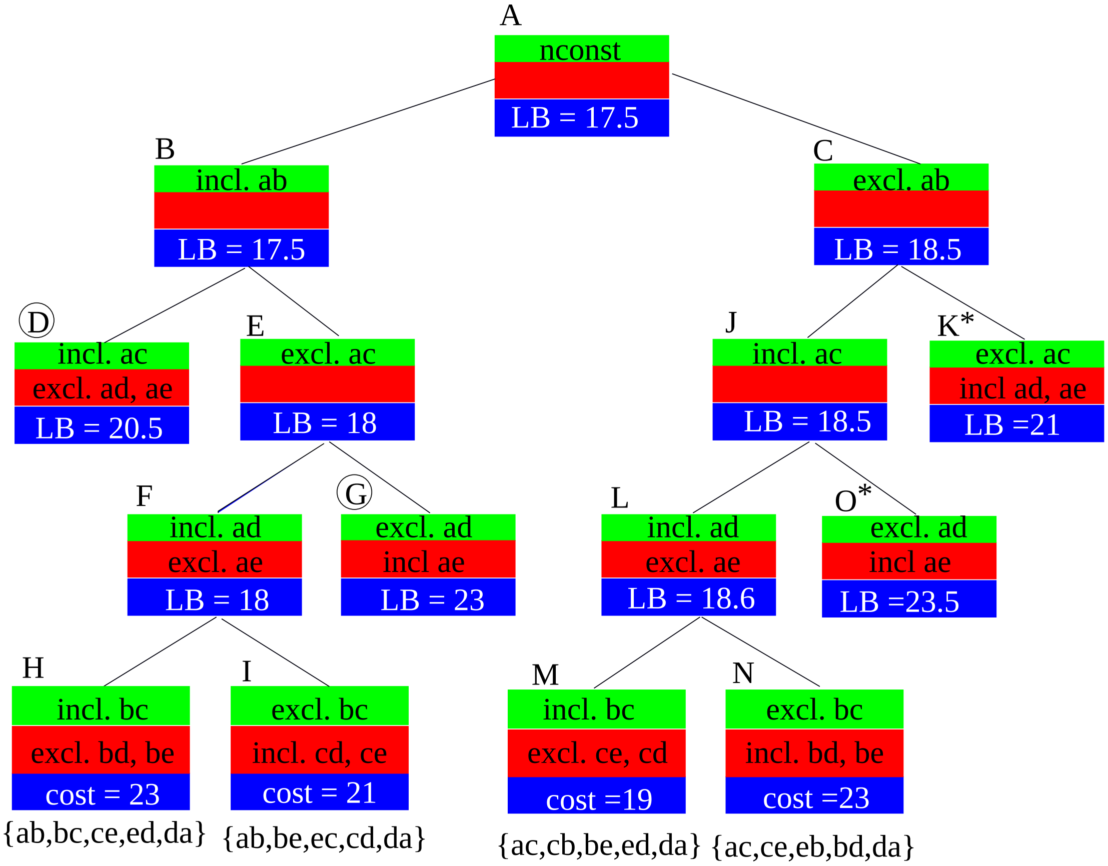
- From the remaining edges at a the lowest cost edges is (a, d). So revise the cost for vertex a as 9.
- (b, c) is included, so from remaining edges at b include (a, b) and revise the cost for $b = 6$.
- From c edge (b, c) is excluded, so of the remaining edges at choose two lowest cost edges (c, a), (c, d) giving the revised cost for c as 9.
- From e edge (a, e) is included. So from the remaining edges the lowest cost edge (b, e) can be chosen giving revised cost $= 10$.
- Hence tour revised bound is $\frac{1}{2}(9+6+9+7+10)$ $= 20$

TSP

- The edges are selected in lexicographic order.
- When a branch has to be taken, determine if edges must be included or excluded on the following basis.
  - If excluding (u, v) makes it impossible for either u or v to have two adjacent edges in the tour then include it.
  - Exclude (u, v) if it raises the degree of either u or v above 2.
  - Exclude (u, v), if it completes a non-tour cycle with the included set of edges.
- After branching compute the lower bound for both children.
- Prune the child whose lower bound is higher than the lowest cost tour found so far.
- If no child can be pruned then continue to explore the child with smaller tour lower bound.

Example: TSP

- Searches from nodes D, G are not continued as cost tour found at I is smaller than lower bounds promised by these nodes.
- Searches from nodes K, O are abandoned as they indicate tour lower bound bigger than lowest cost tour found already at M.



Explanation

- At node H, the constraints are:
  - include: (a, b), (a, d), and (b, c)
  - exclude: (a, c).
- Both edges (b, d) and (b, e) have to be excluded.
- Also both edges (a, c) and (a, e) have to be excluded.
- So, at e edges (e, d) and (e, c) must be included.
- Also at c, edges (c, d) must be included.
- So the tour consists of edges (a, b), (b, c), (c, d), (d, e), (d, a).
- The tour at node I can be explained in a similar way.

Analysis

- Search space is actually a decision tree.
- Each leaf of this decision tree potentially provides a permuation of $n-1$ elements.
- Assuming WLOG that tour begins from city 1, it is of the form $\{1, \pi(\{2, \ldots, n\}), 1\}$.
- Thus, potentially the running time of TSP algorithm can be $O((n-1)!)$.
- But the bounding function in general result in better running time in average.
- So if the bound function is good, there is good chance that the algorithm will avoid many unnecessary search paths, only few nodes of implicit search tree get generated.

