HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

APPLIED ALGORITHMS

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

APPLIED ALGORITHMS

BACKTRACKING, BRANCH AND BOUND

ONE LOVE. ONE FUTURE.

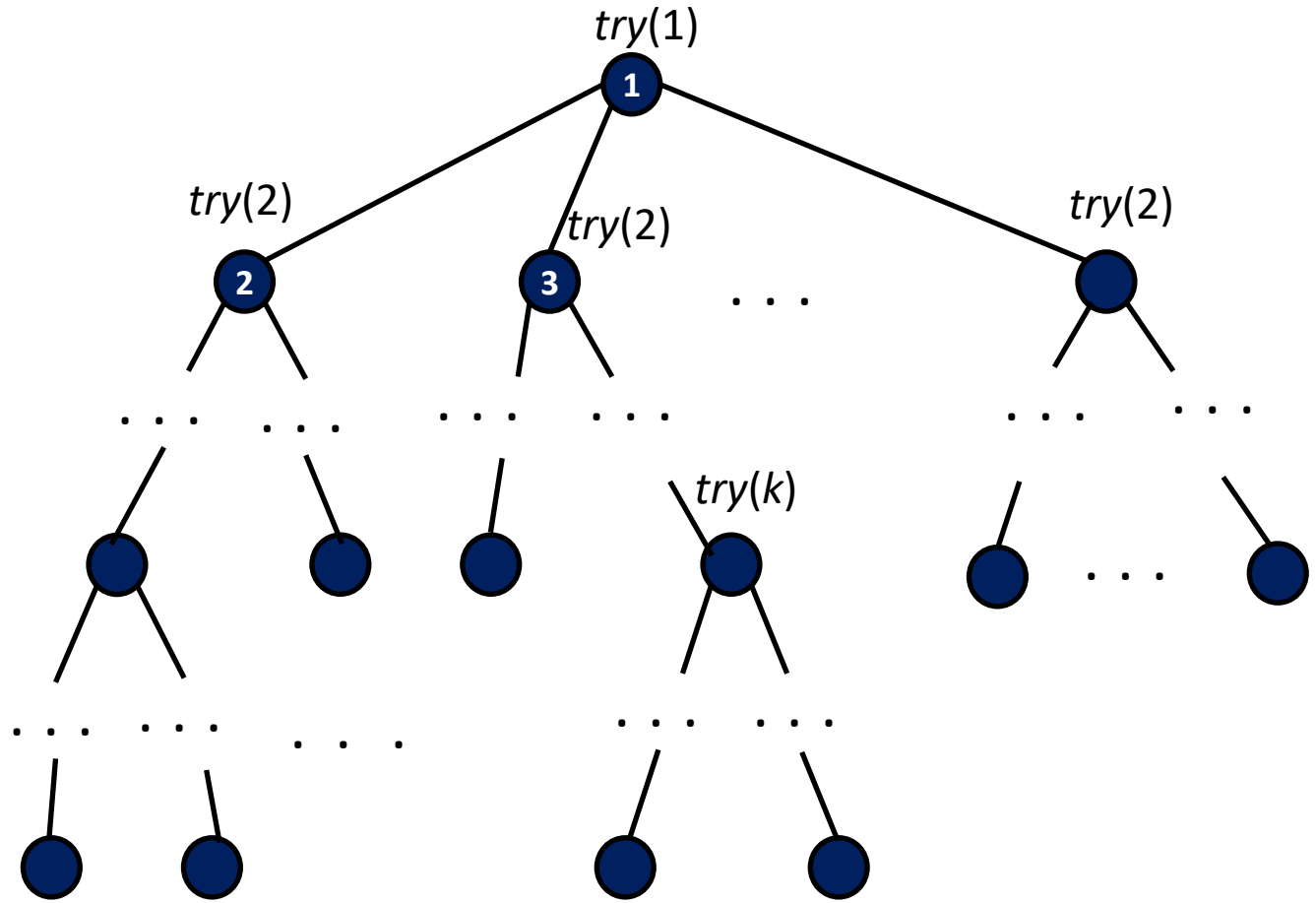
CONTENTS

- General diagram of backtracking, branch and bound
- The problem of bus routes picking up and dropping off passengers
- Delivery truck route problem

GENERAL DIAGRAM OF BACKTRACKING, BRANCHING AND BOUND

- The backtracking algorithm allows us to solve combinatorial enumeration problems and combinatorial optimization problems
- The alternative is modeled by a sequence of decision variables $X_1, X_2, \ldots, X_n$
- Need to find for each variable X_i a value selected from a given discrete set A_i such that
  - The constraints of the problem are satisfied
  - Optimize a given objective function
- Backtracking algorithm
  - Traverse through all variables (e.g. order from $X_1, X_2, \ldots, X_n$), for each variable X_k:
    - Traverse through all possible values that could be assigned to X_k, for each value v:
      - Check constraints
      - Assign $X_k = v$
      - If $k = n$ then record a solution to the problem
      - Otherwise, consider the variable X_{k+1}

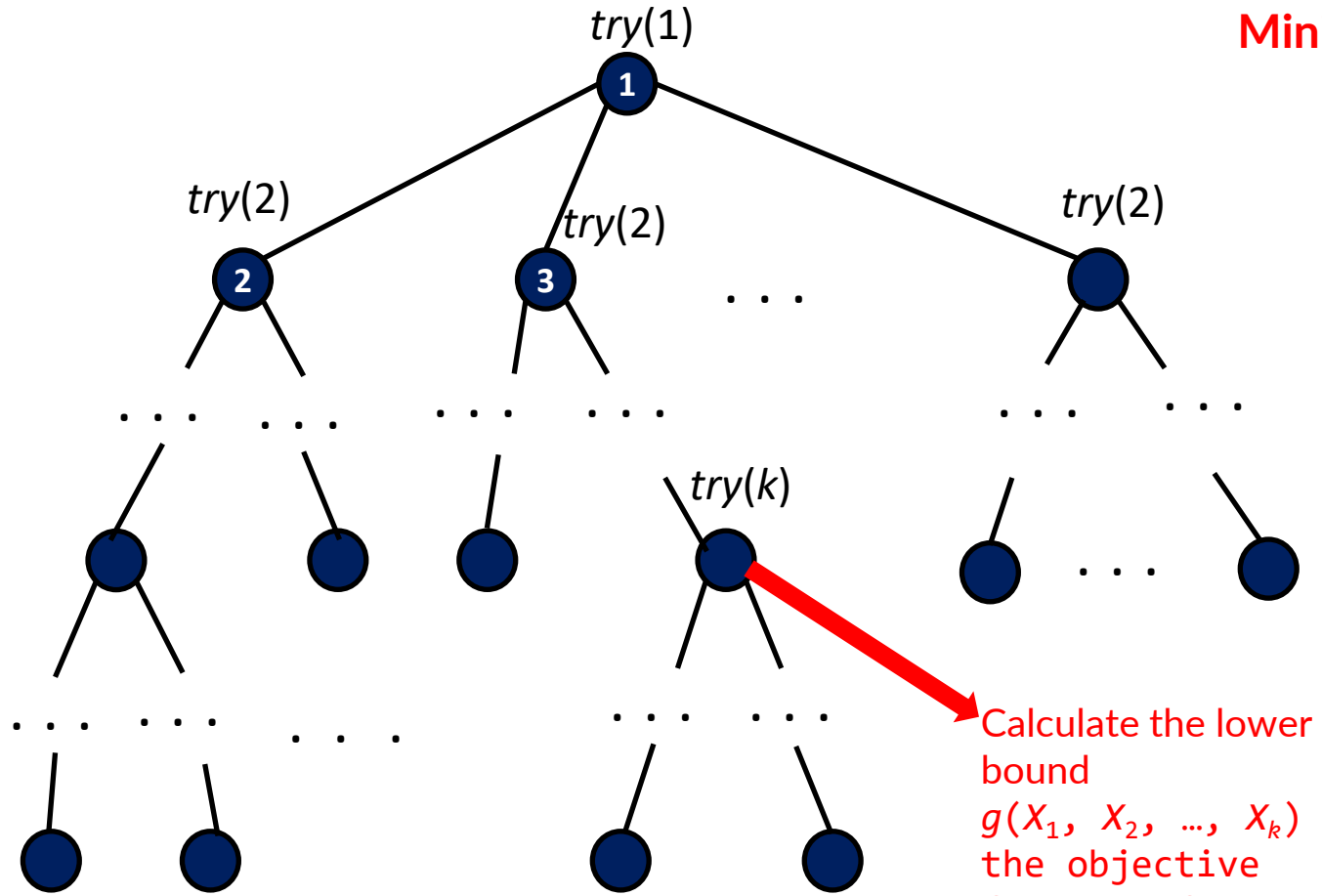
GENERAL DIAGRAM OF BACKTRACKING, BRANCHING AND BOUND



Enumeration problem

```
try(k){ //Try out the possible values assigned to X_k
  for v in A_k do {
     if check(v,k){
        X_k = v;
        [Update a data structure D]
        if k = n then solution();
        else {
            try(k+1);
        }
        [Recover the data structure D]
     }
  }
}
```

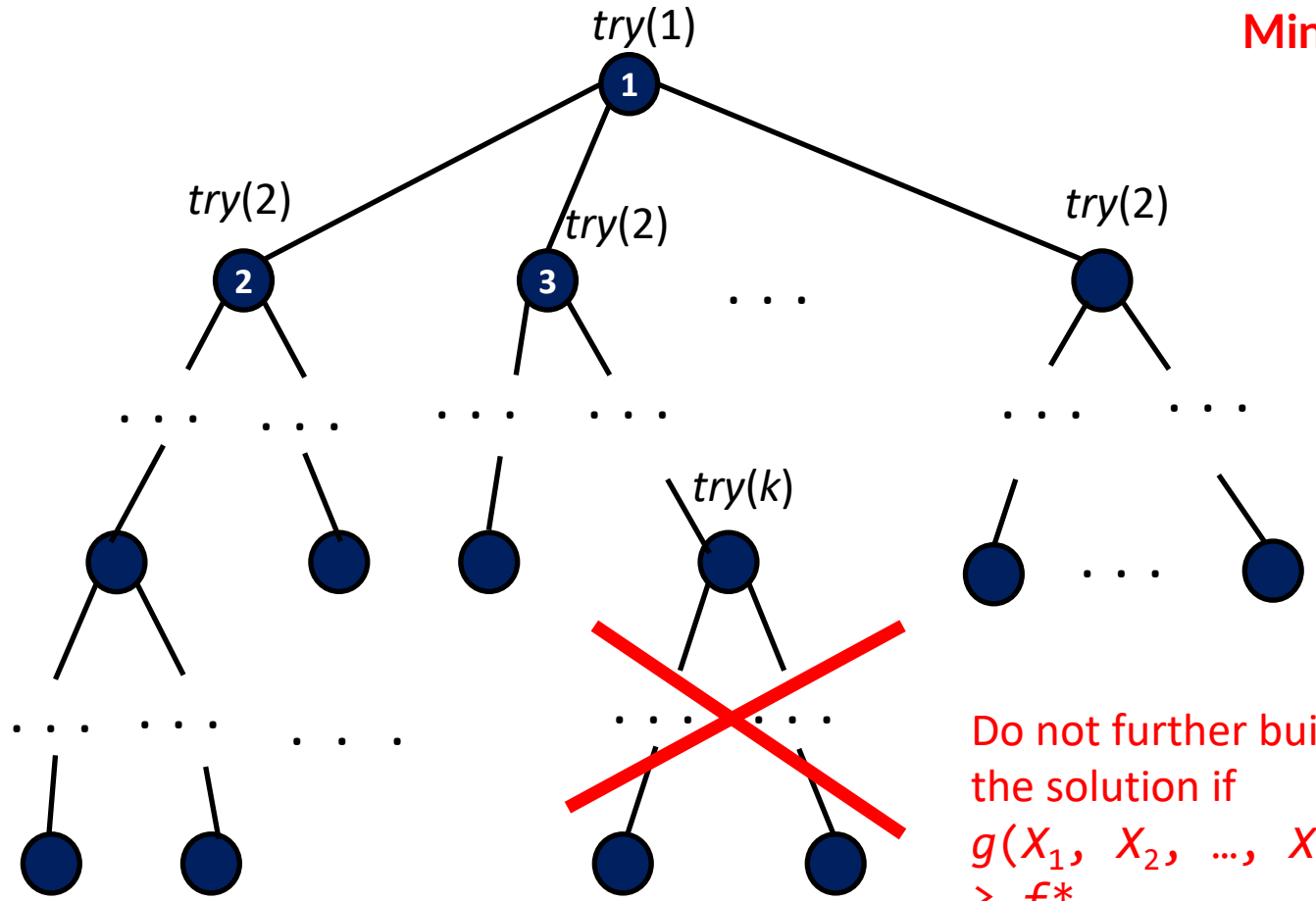
GENERAL DIAGRAM OF BACKTRACKING, BRANCHING AND BOUND



Minimize optimization problem (Denote f^* : optimal value)

```
try(k){//Try out the possible values assigned to X_k
  for v in A_k do {
     if check(v,k){
        X_k = v;
        [Update a data structure D]
        if k = n then updateBest();
        else {
            if g(X_1, X_2, …, X_k) < f* then
                try(k+1);
        }
        [Recover the data structure D]
     }
  }
}
```

GENERAL DIAGRAM OF BACKTRACKING, BRANCHING AND BOUND



Do not further build the solution if $g(X_1, X_2, \ldots, X_k) \geq f^*$

Minimize optimization problem (Denote f^* : optimal value)

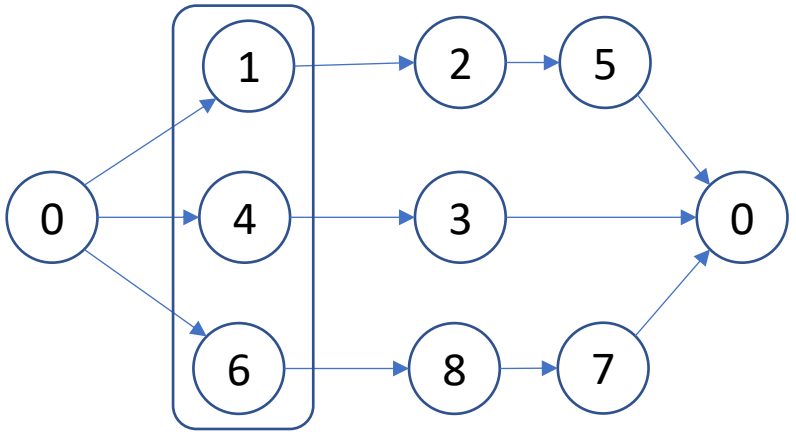
```
try(k){//Try out the possible values assigned to X_k
  for v in A_k do {
     if check(v,k){
        X_k = v;
        [Update a data structure D]
        if k = n then updateBest();
        else {
            if g(X_1, X_2, …, X_k) < f* then
                try(k+1);
        }
        [Recover the data structure D]
     }
  }
}
```

DELIVERY TRUCK ROUTE PROBLEM

- A fleet of *K* identical trucks needs to be assigned to transport pepsi boxes from the central warehouse (point 0) to delivery points 1,2,...,*N* and back to the warehouse. The travel distance from point *i* to point *j* is *c*(*i*,*j*)
- Each truck has a load capacity of Q (each trip can only transport a maximum of Q boxes)
- Each delivery point *i* has a required quantity of *d*[*i*] boxes, *i* = 1,..., *N*.
- It is necessary to develop a transportation plan so that:
  - Each delivery point can only be delivered by exactly one vehicle and only once.
  - The total amount of boxes on the vehicle must not exceed the vehicle's load.
  - The total route length of the vehicles is the smallest
- Note: do not need to use all K vehicles.

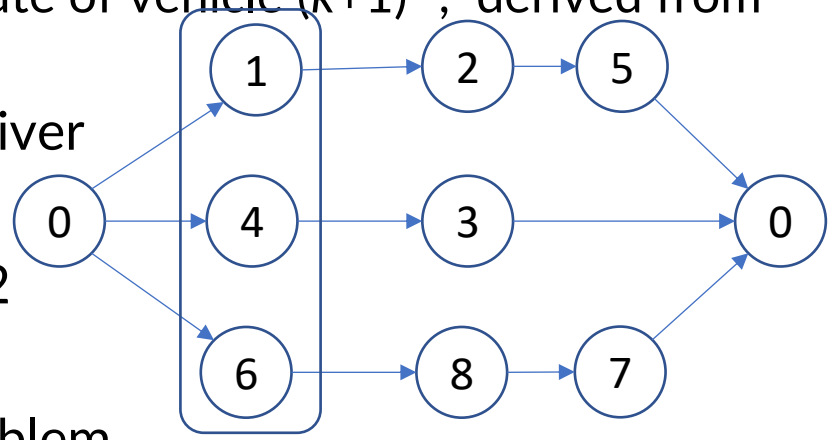
DELIVERY TRUCK ROUTE PROBLEM

- Design data
  - $y[k]$ first delivery point of the *kth* vehicle ($y[k] \in \{0, 1, 2,..., N\}$, where $k = 1, 2,..., K$)
    - $y[k] = 0$ means the vehicle k will not be used for route planning
  - $x[i]$ is the next point of the delivery point i on the route ($x[i] \in \{0,1,2,...,N\}$, với $i = 1, 2,..., N$)
  - Since the vehilces are identical, we can assume $y[k] \leq y[k+1]$, $k = 1, 2,..., K-1$
    - If $y[k] > 0$ then $y[k+1] > y[k]$
  - Variables associated with the partial solution:
    - visited[v] = true if v has been visited by a vehicle
    - load[k]: total amount of boxes on the vehicle k
    - f: total length of the current partial solution
    - f^*: best distance that has been found



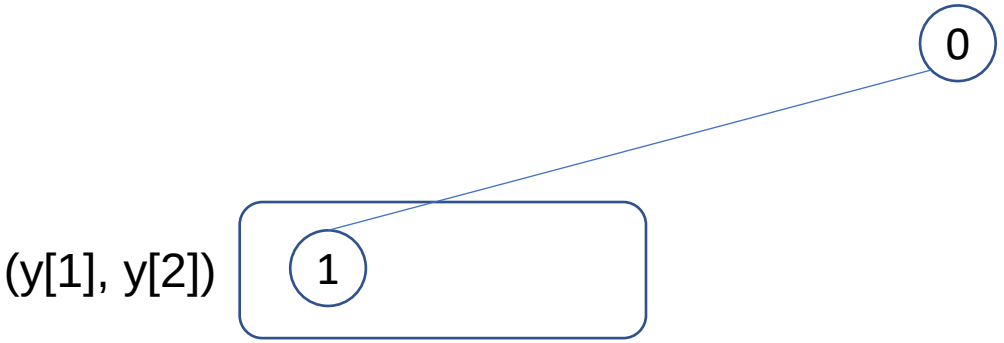
DELIVERY TRUCK ROUTE PROBLEM

- Strategy to traverse
  - Start by traversing values for the tuple ($y[1],. . ., y[K]$)
  - For each complete values of tuple ($y[1],. . ., y[K]$), start traversing values for the $x[1,...,N]$ derived f rom $x[y[1]]$
  - Each time: try to assign $x[v] = u$ for the kth vehicle:
    - If $u > 0$ (not the starting point): try continuing to browse the value for $x[u]$ still on the kth vehicle
    - If $u = 0$ (starting point) then
      - If $k = K$ (all routes for K vehicles are complete) and all delivery points are visited, then obtain a solution to the problem
      - otherwise, try continuing to assign the values for the route of vehicle $(k+1)^{th}$, derived from $x[y[k+1]]$
  - Variable nbR: the number of vehicles that has been used to deliver
  - Variable segments
    - The number of segments (segment: connection between 2 consecutive delivery points on the route)
    - when segments = N+nbR then obtain a solution to the problem



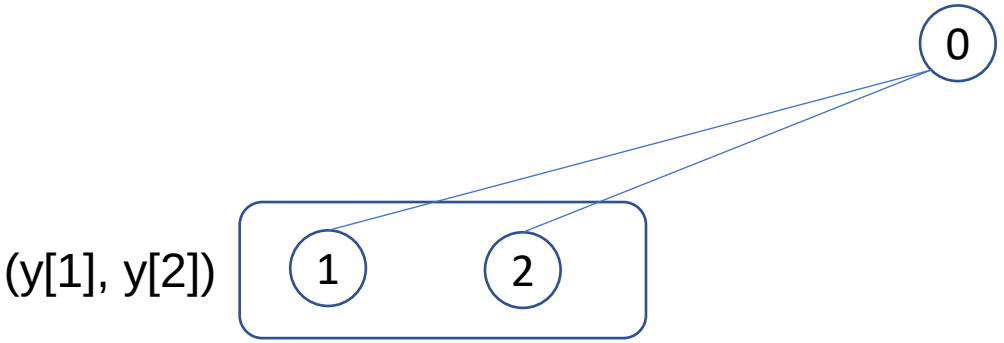
DELIVERY TRUCK ROUTE PROBLEM

- K = 2, N= 6



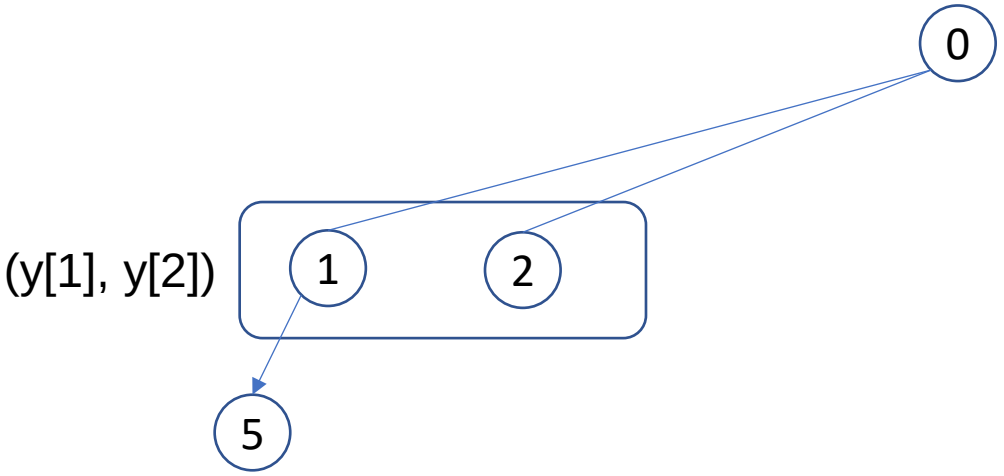
DELIVERY TRUCK ROUTE PROBLEM

- K = 2, N= 6



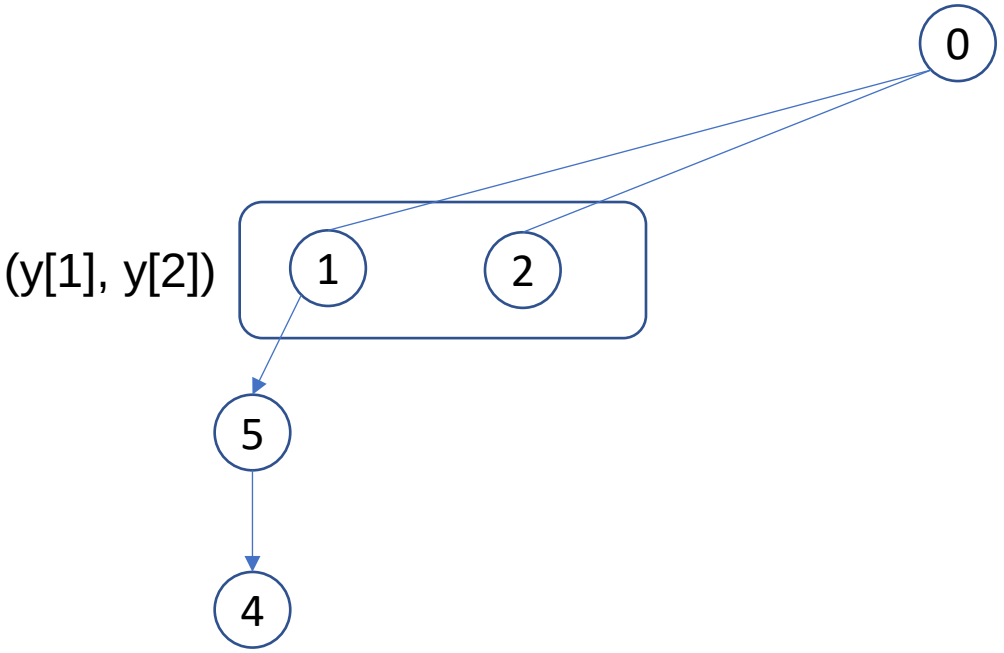
DELIVERY TRUCK ROUTE PROBLEM

- K = 2, N= 6



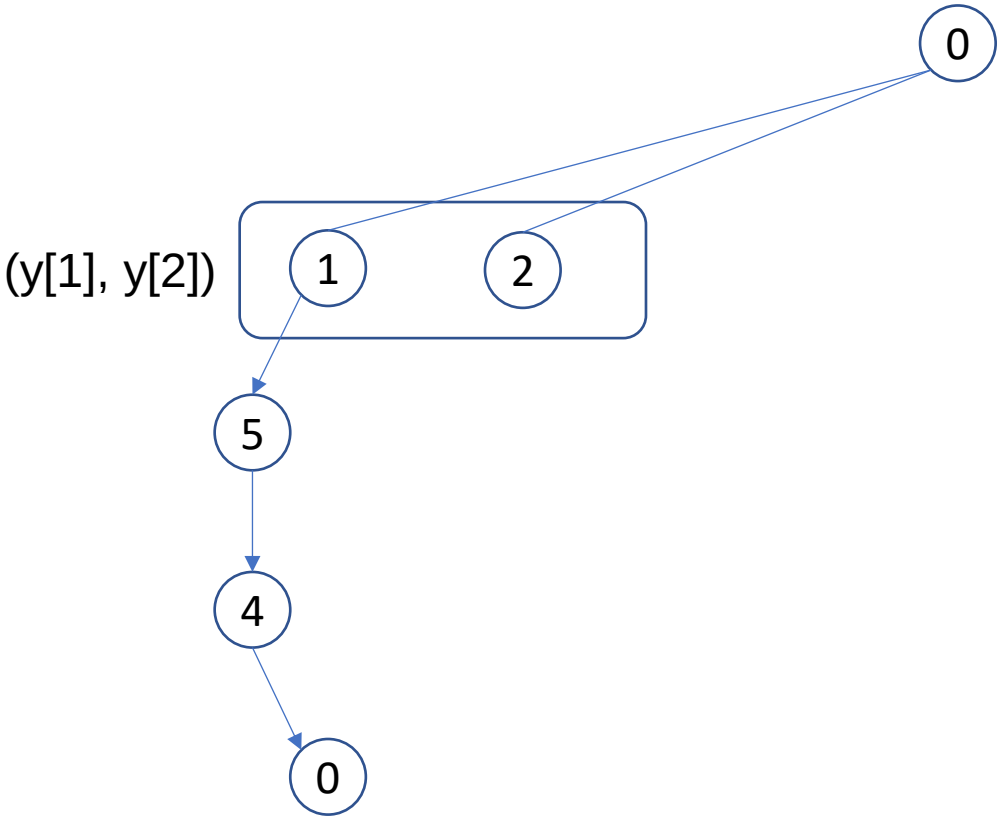
DELIVERY TRUCK ROUTE PROBLEM

- K = 2, N= 6



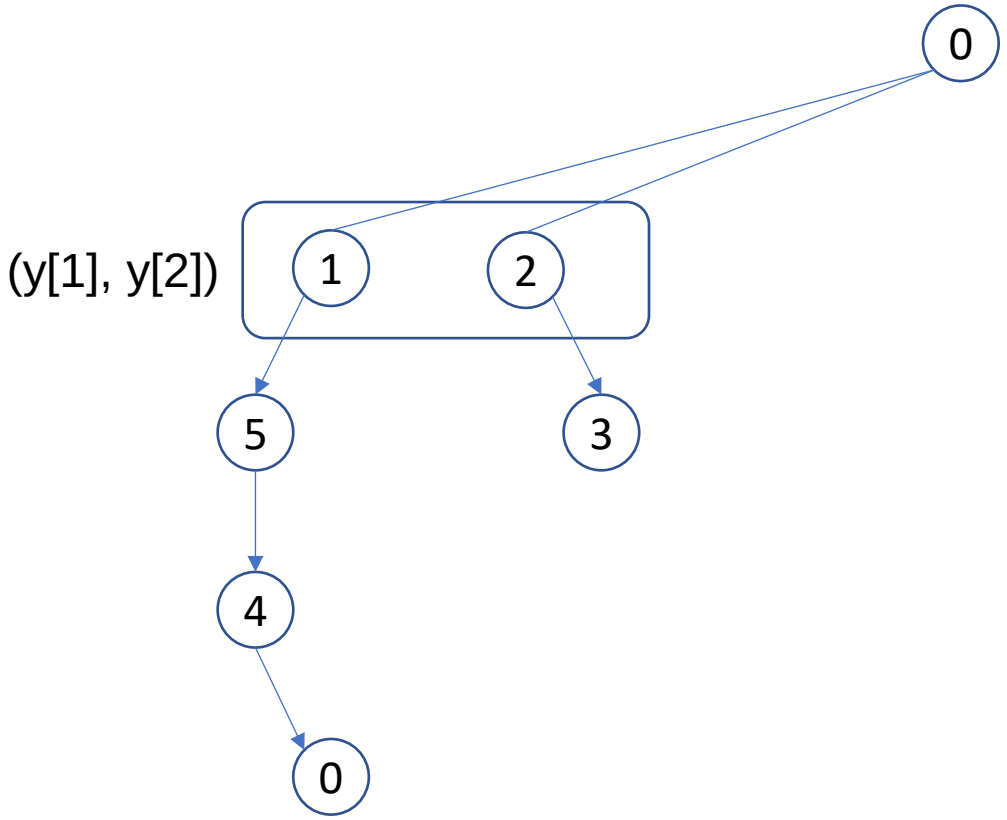
DELIVERY TRUCK ROUTE PROBLEM

- K = 2, N= 6



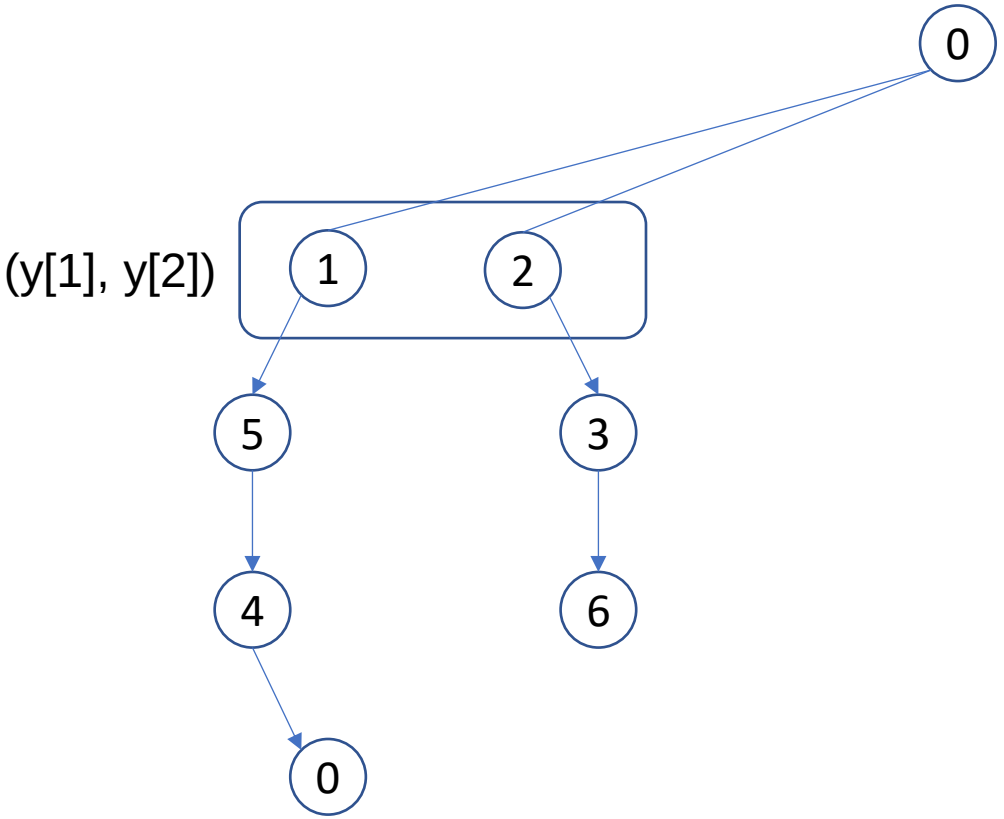
DELIVERY TRUCK ROUTE PROBLEM

- K = 2, N= 6



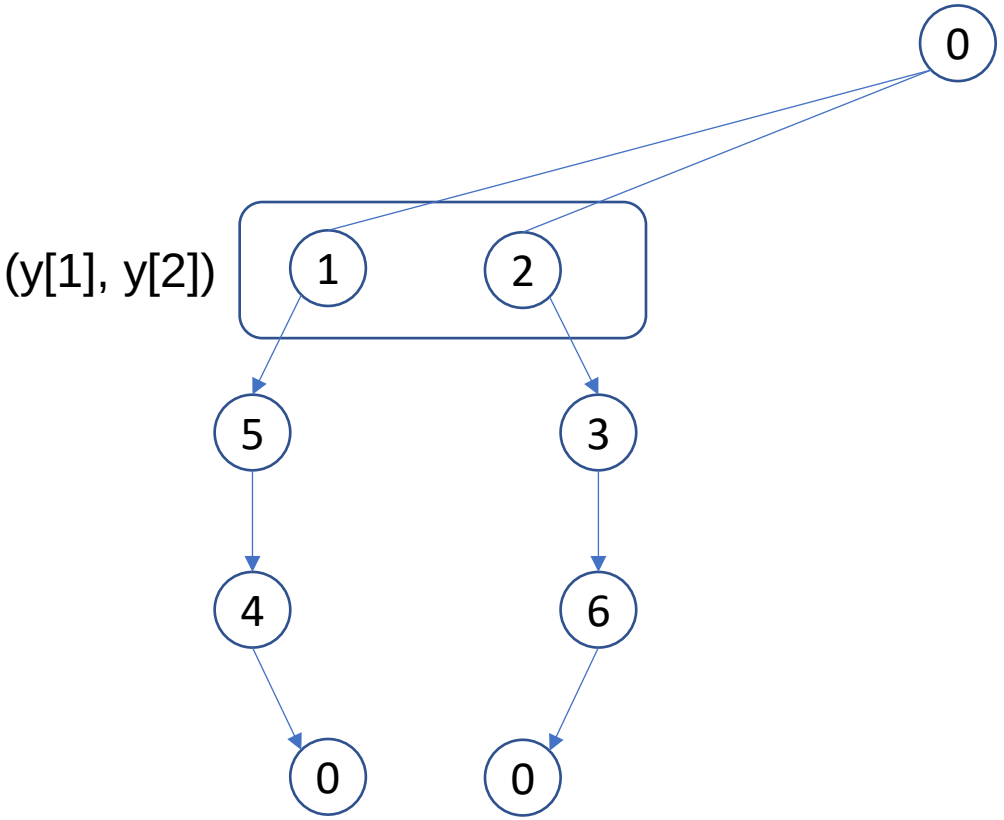
DELIVERY TRUCK ROUTE PROBLEM

- K = 2, N= 6



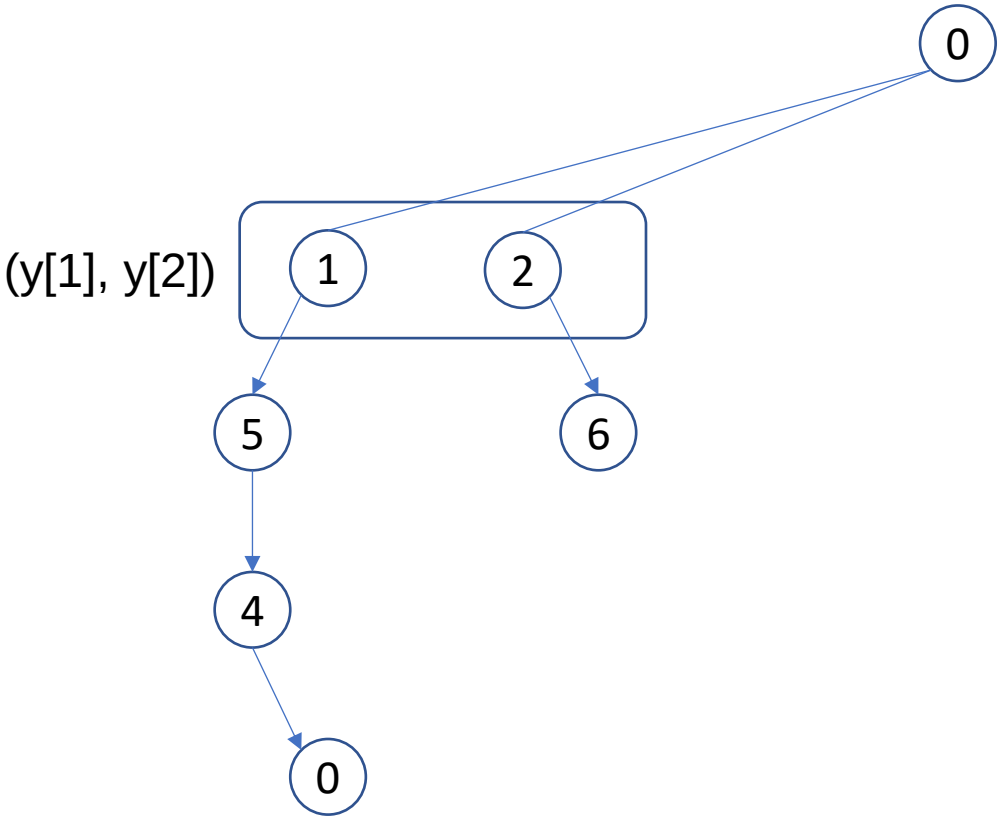
DELIVERY TRUCK ROUTE PROBLEM

- K = 2, N= 6



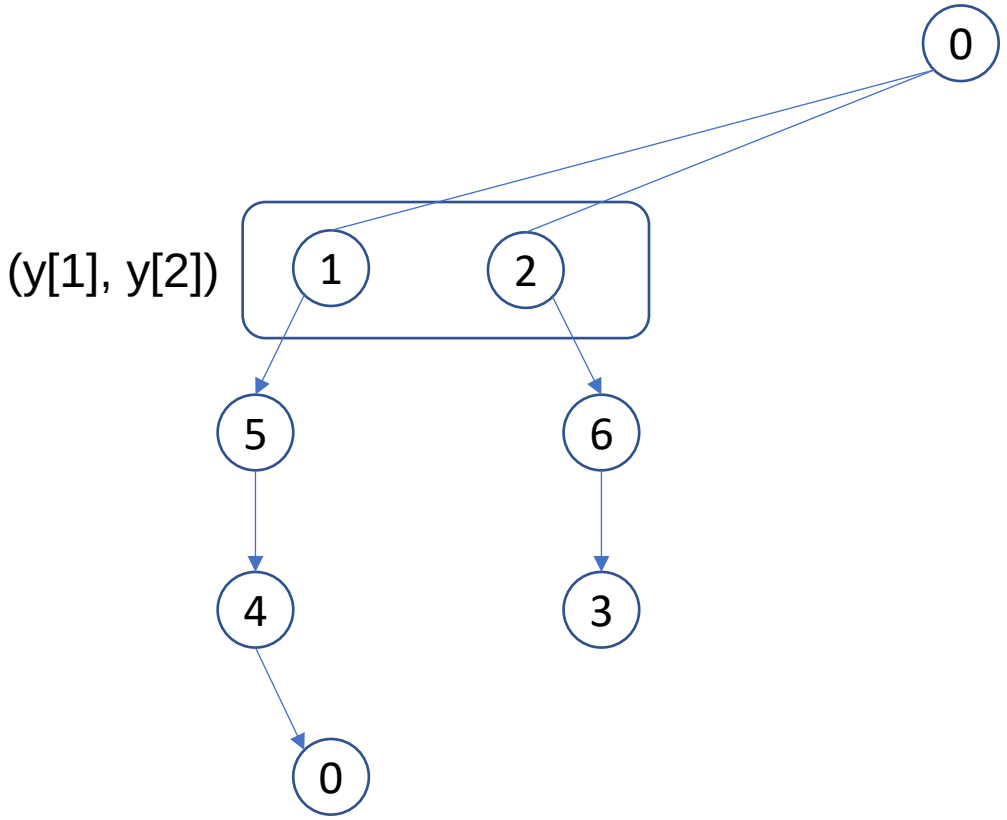
DELIVERY TRUCK ROUTE PROBLEM

- K = 2, N= 6



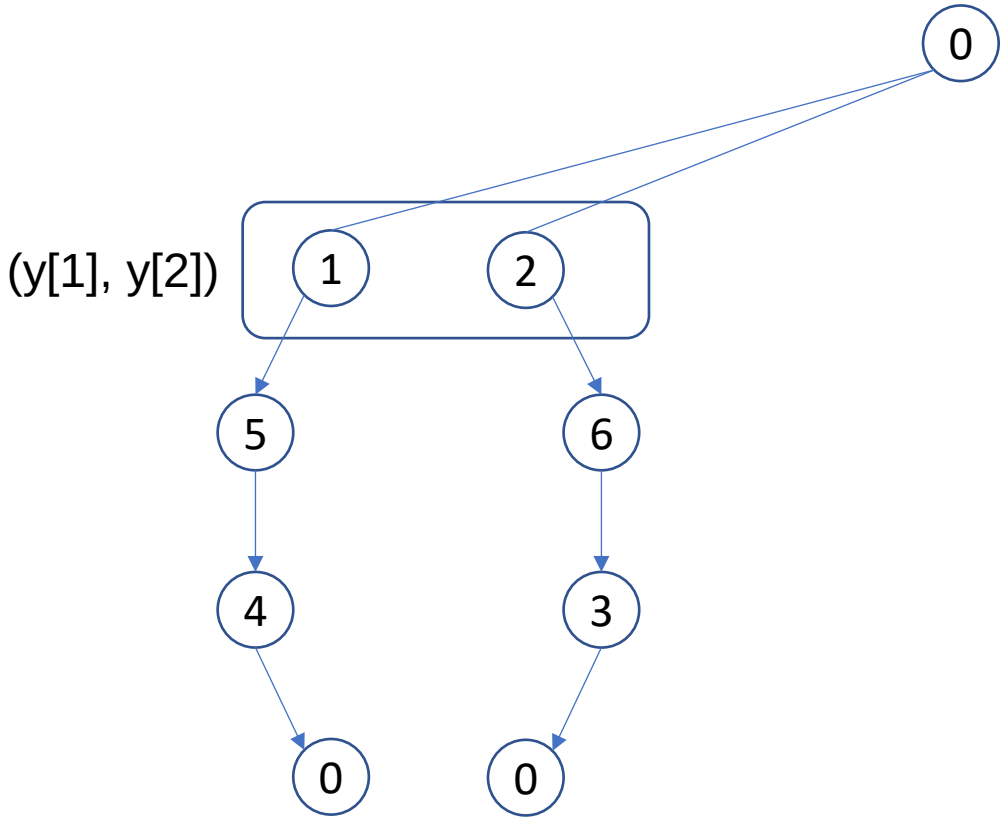
DELIVERY TRUCK ROUTE PROBLEM

- K = 2, N= 6



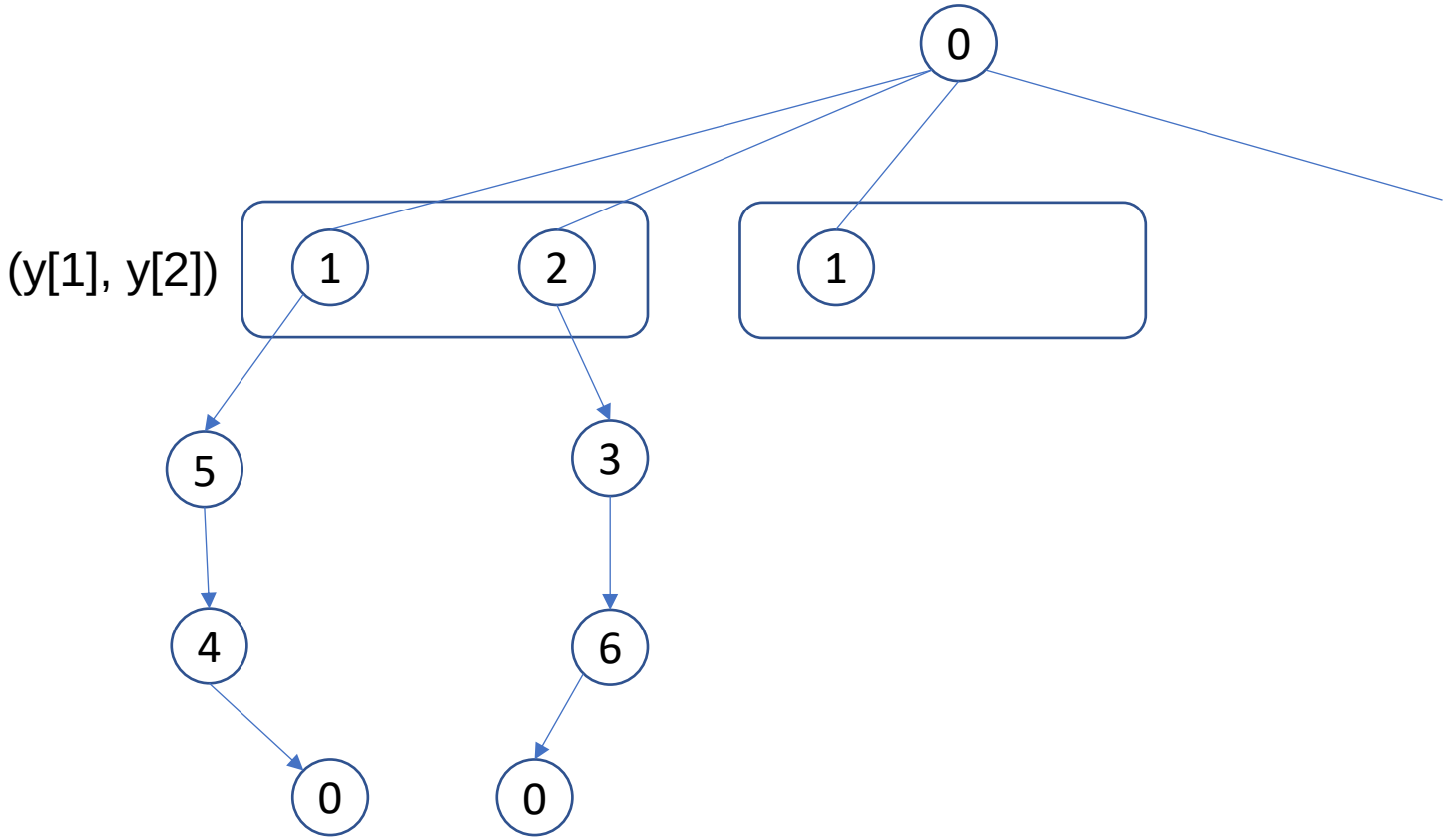
DELIVERY TRUCK ROUTE PROBLEM

- K = 2, N= 6



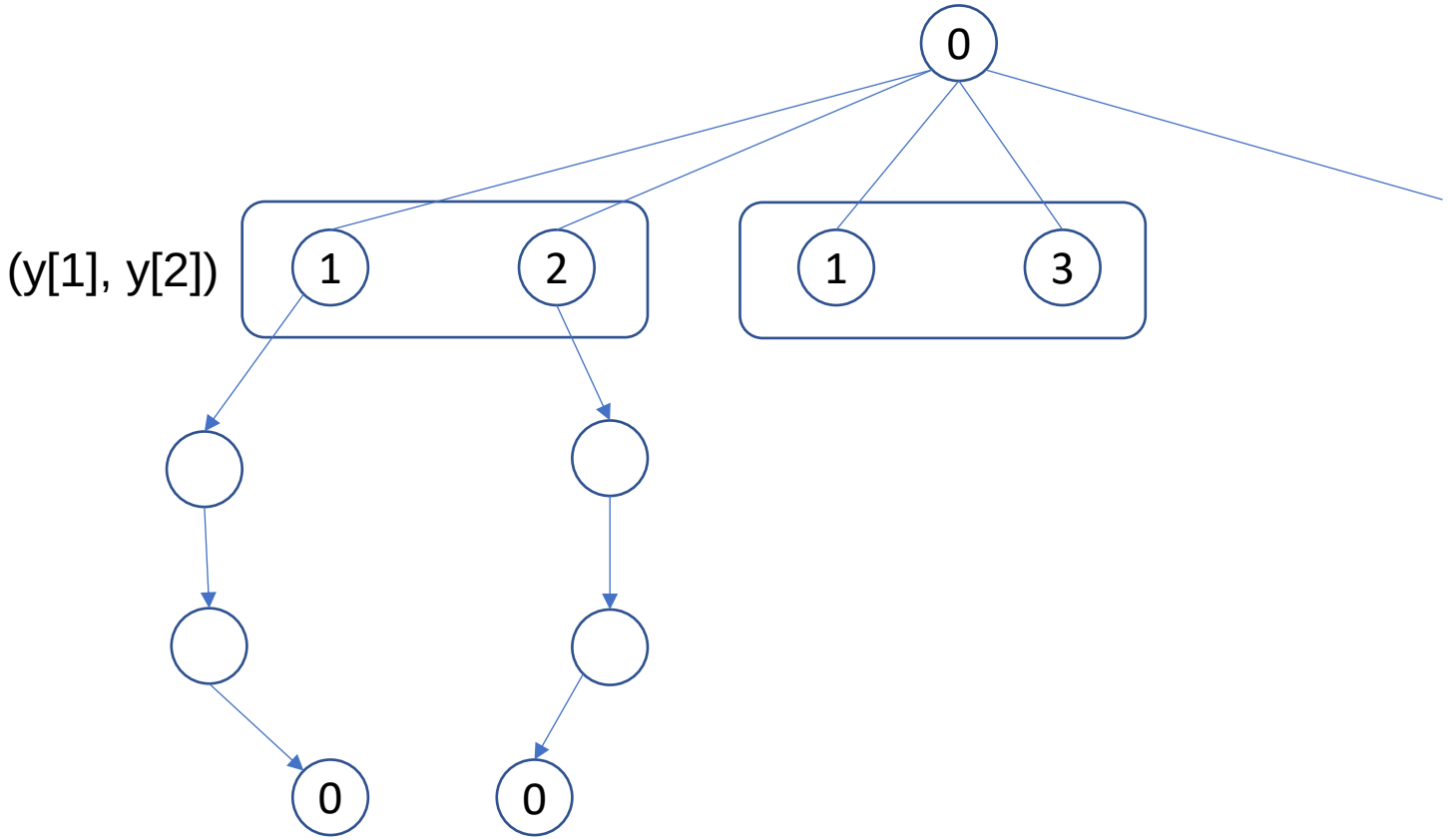
DELIVERY TRUCK ROUTE PROBLEM

- K = 2, N= 6



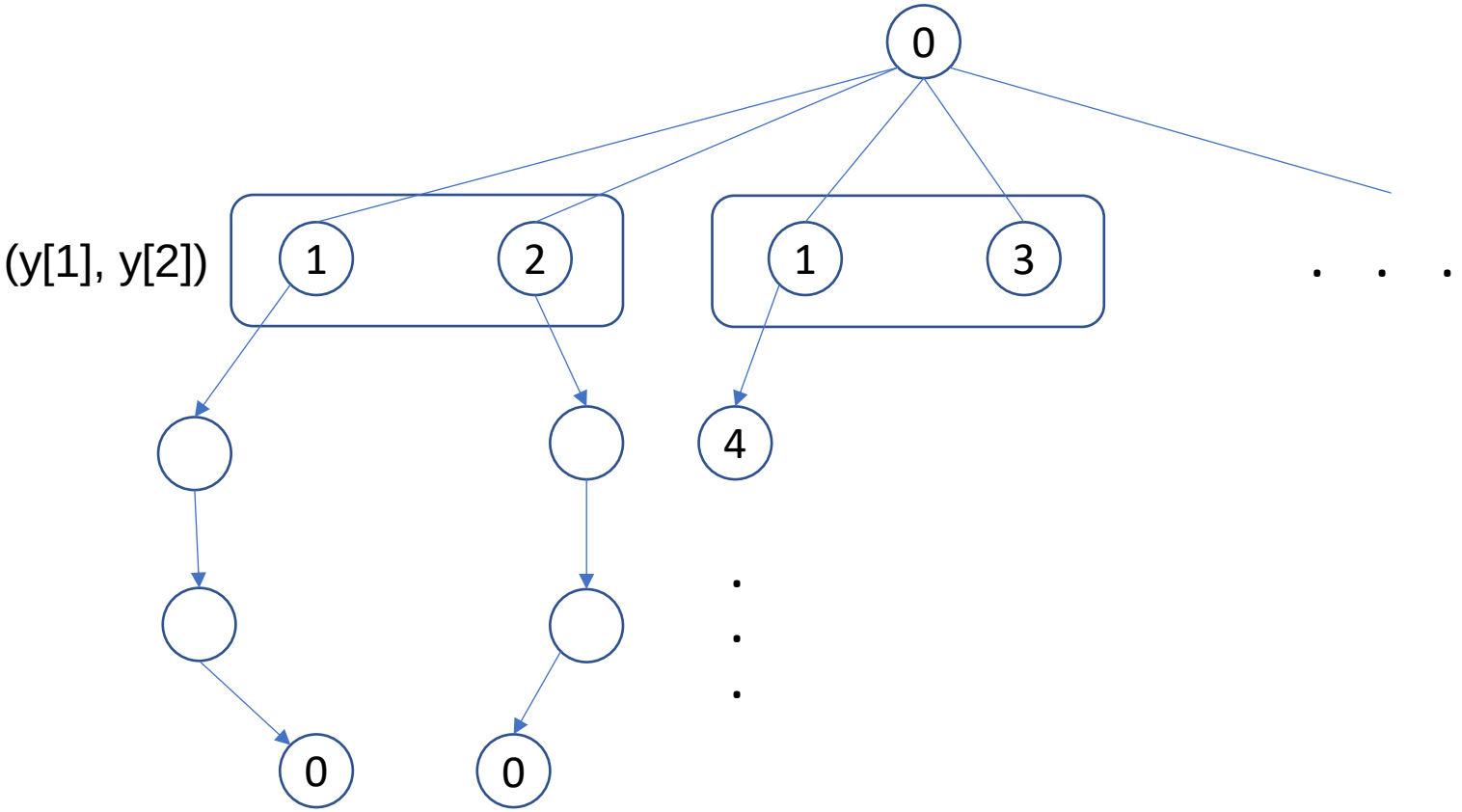
DELIVERY TRUCK ROUTE PROBLEM

- K = 2, N= 6



DELIVERY TRUCK ROUTE PROBLEM

- K = 2, N= 6



DELIVERY TRUCK ROUTE PROBLEM

```
TRY_X(s, k){// thử giá trị cho x[s]
  if(s = 0) then{
    if k < K then
       TRY_X(y[k+1],k+1);
    return
  }
  [. . .]
}
```

```
checkX(v, k){
   if v > 0 and visited[v]
        then return false;
   if load[k] + d[v] > Q
        then return false;
   return true;
}
```

```
for v = 0 to n do {
    if checkX(v,k) then {
      x[s] = v; visited[v] = true; f = f + c[s,v];
      load[k] = load[k] + d[v]; segments = segments + 1;
      if v > 0 then { if f + (n + nbR – segments)*Cmin < f* then TRY_X(v,k); }
      else{
        if k = K then {
          if segments = n + nbR then updateBest();
        }else{
          if f + (n + nbR – segments)*Cmin < f* then  TRY_X(y[k+1],k+1);
        }
      }
      visited[v] = false; f = f – c[s,v];
      load[k] = load[k] – d[v]; segments = segments – 1;
    }
}
```

DELIVERY TRUCK ROUTE PROBLEM

```
checkY(v, k){
   if v = 0 then return true;
   if load[k] + d[v] > Q then
     return false;
   if visited[v] = true then
     return false;
   return true;
}
```

```
solve(){
  f = 0; f* = +∞; y[0] = 0;
  for v = 1 to n do
    visited[v] = false;
  TRY_Y(1);
  output(f*);
}
```

```
TRY_Y(k){ // thử giá trị cho y[k]
   s = 0;
   if y[k-1] > 0 then s = y[k-1] + 1;
   for v = s to n do {
     if checkY(v,k) then  {
        y[k] = v;
        if v > 0 then segments = segments + 1;
        visited[v] = true;  f = f + c[0,v]; load[k] = load[k] + d[v];
        if k < K then TRY_Y(k+1);
        else { nbR = segments; TRY_X(y[1],1);   }
        load[k] = load[k] – d[v]; visited[v] = false; f = f – c[0,v];
        if v > 0 then segments = segments – 1;
     }
   }
}
```

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !