

Minimum cost spanning r -arborescence

Combinatorial Optimization

Giovanni Righini



The problem

Problem data:

- a digraph $\mathcal{D} = (\mathcal{N}, \mathcal{A})$,
- a node $r \in \mathcal{N}$,
- a cost function $c : \mathcal{A} \rightarrow \mathbb{R}_+$.

Problem (Minimum Cost Spanning r -Arborescence Problem).

Find a spanning r -arborescence of minimum cost.

A digraph $\mathcal{T} = (\mathcal{N}, \mathcal{A})$ is a **spanning rooted arborescence** (r -arborescence, for short) if and only if there is a unique directed path from its root node $r \in \mathcal{N}$ to all the other nodes in $\mathcal{N} \setminus \{r\}$ and no directed path from any node in $\mathcal{N} \setminus \{r\}$ to r .



Counter-example

The algorithms for the MSTP do not work.

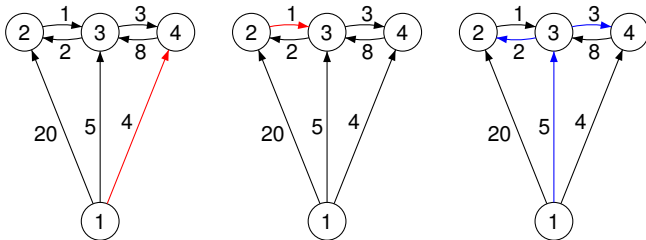


Figure: Prim algorithm would select (1, 4) first. Kruskal algorithm would select (2, 3) first. None of them belongs to the optimal solution.



A mathematical programming model

$$\begin{aligned} \min z &= \sum_{a \in \mathcal{A}} c_a x_a \\ \text{s.t. } \sum_{a \in \delta(S)} x_a &\geq 1 & \forall S \subseteq \mathcal{N} \setminus \{r\} \\ x_a &\in \{0, 1\} & \forall a \in \mathcal{A} \end{aligned} \quad (1)$$

A binary variable x_a indicates whether each arc $a \in \mathcal{A}$ belongs to the solution or not.

We call r -cuts all arc subsets $\delta(S)$ corresponding to all node subsets S not containing r , made of the arcs entering S :

$$\delta(S) = \{(i, j) \in \mathcal{A} : i \notin S \wedge j \in S\} \quad \forall S \subset \mathcal{N} : r \notin S$$

Constraints (1) impose that all such subsets be reached by at least one entering arc.

Integrality conditions are redundant. We initially assume this and then we prove that the optimal solution obtained from the linear relaxation is integer.



A mathematical programming model

The linear relaxation is

$$\begin{aligned} \min z &= \sum_{a \in \mathcal{A}} c_a x_a \\ \text{s.t. } \sum_{a \in \delta(S)} x_a &\geq 1 & \forall S \subseteq \mathcal{N} \setminus \{r\} \\ 0 \leq x_a &\leq 1 & \forall a \in \mathcal{A}. \end{aligned}$$

- All terms of the sums $\sum_{a \in \delta(S)} x_a$ are non-negative;
- the right-hand-side is equal to 1.

Therefore, no x variable is required to take value larger than 1 to achieve **feasibility**.

The objective function coefficients c are non-negative.

Therefore, no x variable is required to take value larger than 1 to achieve **optimality**.

Therefore, the bounds $x_a \leq 1$ are redundant.



The primal-dual pair

The **primal problem**.

$$\begin{aligned} \min z &= \sum_{a \in \mathcal{A}} c_a x_a \\ \text{s.t.} \quad &\sum_{a \in \delta(S)} x_a \geq 1 && \forall S \subseteq \mathcal{N} \setminus \{r\} \\ &x_a \geq 0 && \forall a \in \mathcal{A} \end{aligned}$$

The **dual problem**.

$$\begin{aligned} \max w &= \sum_{S \subseteq \mathcal{N} \setminus \{r\}} y_S \\ \text{s.t.} \quad &\sum_{S \subseteq \mathcal{N} \setminus \{r\} : a \in \delta(S)} y_S \leq c_a && \forall a \in \mathcal{A} \\ &y_S \geq 0 && \forall S \subseteq \mathcal{N} \setminus \{r\} \end{aligned}$$



Complementary slackness conditions

Primal constraints and **dual variables** correspond to subsets S not including r , i.e. to r -cuts.

Primal C.S.C.: $x_a(c_a - \sum_{S \subseteq \mathcal{N} \setminus \{r\}: a \in \delta(S)} y_S) = 0 \quad \forall a \in \mathcal{A}$

Dual C.S.C.: $y_S(\sum_{a \in \delta(S)} x_a - 1) = 0 \quad \forall S \subseteq \mathcal{N} \setminus \{r\}$

The initial **primal solution** is $x_a = 0 \quad \forall a \in \mathcal{A}$ and it is **primal infeasible** (and super-optimal).

The initial **dual solution** is $y_S = 0 \quad \forall S \subseteq \mathcal{N} \setminus \{r\}$ and it is **dual feasible** (and sub-optimal).



Complementary slackness conditions

Owing to the constraints

$$\sum_{a \in \delta(S)} x_a \geq 1 \quad \forall S \subseteq \mathcal{N} \setminus \{r\}$$

primal infeasibility is measured by the number of unreached node subsets (SCCs).

Owing to the primal C.S.C.

$$x_a \left(c_a - \sum_{S \subseteq \mathcal{N} \setminus \{r\} : a \in \delta(S)} y_S \right) = 0 \quad \forall a \in \mathcal{A}$$

the only arcs that can be used to traverse r -cuts are those with null reduced cost (**admissible arcs**):

$$x_a = 1 \Rightarrow \bar{c}_a = 0$$

where $\bar{c}_a = c_a - \sum_{S \subseteq \mathcal{N} \setminus \{r\} : a \in \delta(S)} y_S$.



Edmonds algorithm

This algorithm is due to Chu and Liu (1965), Edmonds (1967), Bock (1971). It is a primal-dual algorithm. Iteratively it does the following:

- **Dual iteration**: select a **violated primal constraint**. It corresponds to an **r -cut** that does not contain any admissible arc: its correspondent **SCC S** is not reached from r through admissible arcs. Make the corresponding **dual variable y_S** basic: it is increased (**dual ascent procedure**) until it activates one or more **dual constraints** corresponding to one or more **primal variables x_a** (arcs). Such arcs get zero reduced cost and therefore they become **admissible**.
- **Primal iteration**: select a **primal variable x_a** corresponding to an admissible arc entering **S** and make it basic: increase it to repair the infeasibility of the selected **primal constraint**. The **r -cut** corresponding to **S** is now traversed by the selected **basic arc a** .



Two approaches

In the scientific and educational literature Edmonds algorithm is presented in two different ways.

- **Description 1:**

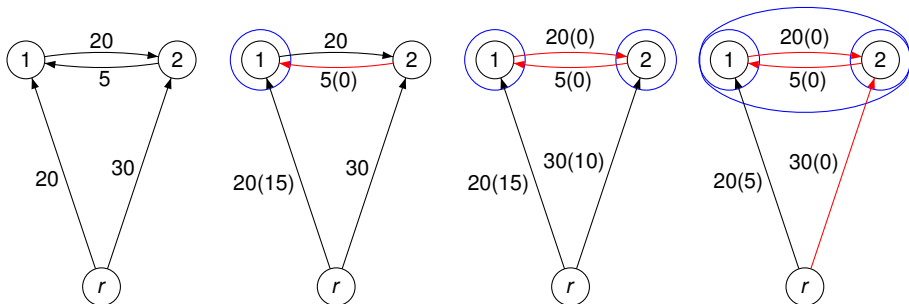
- no reference to duality theory, no **dual data-structure**;
- the a priori knowledge of r is assumed;
- SCCs not reachable from r are identified by a sequence of calls to a graph search sub-routine (such as DFS or BFS);
- complexity: $O(mn)$.

- **Description 2:**

- Step 1: build a **dual data-structure**;
- Step 2: find the optimal solution;
- the construction of the dual data-structure is independent of r ;
- better computational complexity, such as $O(m \log n)$, depending on the data-structure used;
- sophisticated implementations are possible.



Example



To retrieve a corresponding optimal primal solution we need to select some of the admissible arcs as basic (some others can be redundant).

In this example there are three admissible arcs: arcs $(r, 2)$ and $(2, 1)$ are basic; arc $(1, 2)$ is redundant.



Edmonds algorithm: correctness (3)

Redundant arcs are identified by possibly removing them following the reverse insertion order: the latest inserted arc survives among those with the same head.

After removing redundant arcs, no two arcs entering the same node $v \in \mathcal{N} \setminus \{r\}$ can remain and all cycles (SCCs) are broken.

All basic variables have value 1.

Therefore the final primal solution is made of $n - 1$ arcs and therefore it is an r -arborescence.



Edmonds algorithm: complexity (1)

Edmonds algorithm can be implemented with complexity $O(nm)$, because it requires at most $2(n-1)$ iterations and each of them has complexity $O(m)$.

Let h be the number of maximal SCCs of $\mathcal{D}_B$ (excluding $\{r\}$).

Let h_0 be the number of maximal SCCs of $\mathcal{D}_B$ with no admissible entering arcs (excluding $\{r\}$).

At each iteration, the sum $h + h_0$ decreases by at least 1:

- if the selected SCC S remains a SCC, then it gets an entering admissible arc: therefore, h does not change and h_0 decreases;
- if S is merged with another SCC, then h decreases and h_0 does not increase.

Initially $h = n - 1$ and $h_0 = n - 1$. Eventually $h \geq 0$ and $h_0 = 0$.

Therefore the iterations are at most $2(n-1)$.



Edmonds algorithm: complexity (2)

In $O(m)$ time it is possible to find a SCC S in $\mathcal{D}_B$, sorting the nodes in pre-topological order, starting from r , so that the first node belongs to a SCC not reachable from r and with no admissible entering arcs.

Therefore each iteration has time complexity $O(m)$.

In this implementation

- the root r must be known since the beginning;
- more than one arc may become admissible in a same iteration;
- no dual data-structure is required;
- any graph search algorithm can be used as a sub-routine (DFS, BFS,...).



Iterative dual ascent

for $a \in \mathcal{A}$ **do**

$\bar{c}_a \leftarrow c_a$

$Order \leftarrow VisitGraphFw$

while $Order[n] \neq r$ **do**

$S \leftarrow VisitGraphBw(Order[n])$

$\alpha \leftarrow FindArc(S)$

$UpdateCost(S, \alpha)$

$Order \leftarrow VisitGraphFw$

Upon termination, a vector of predecessors, $Pred$, identifies the
primal optimal solution.



Procedure *VisitGraphFw*

```
for  $i \in \mathcal{N}$  do  
     $Pred[i] \leftarrow nil$   
 $Pred[0] \leftarrow 0$   
 $k \leftarrow 0$   
DFS(0)
```

Node 0 is a dummy node connected to all others with null cost arcs $(0, i) \forall i \in \mathcal{N}$.

The vector $Pred$ indicates the predecessors when the digraph $\mathcal{D}_B$ is visited.

The vector $Order$ indicates the reverse pre-topological order of the nodes.

k is the index of $Order$.



Procedure DFS(*i*)

```
for (i, j) ∈  $\delta^+(i)$  do  
    if ( $\overline{c}_{ij} = 0$ ) ∧ (Pred[j] = nil) then  
        Pred[j] ← i  
        DFS(j)  
k ← k + 1  
Order[k] ← i
```

DFS is a recursive procedure.



Procedure *VisitGraphBw*

```
for  $i \in \mathcal{N}$  do  
     $Succ[i] \leftarrow nil$   
     $Succ[Order[n]] \leftarrow Order[n]$   
     $h \leftarrow 0$   
     $RevDFS(Order[n])$ 
```

The vector *Succ* indicates the successors when the graph is visited backward.

S is the characteristic vector of a strongly connected component.

h is its index.



Procedure RevDFS(*i*)

```
for ( $j, i$ )  $\in \delta^-(i)$  do  
  if ( $\overline{c}_{ji} = 0$ )  $\wedge$  ( $Succ[j] = nil$ ) then  
     $Succ[j] \leftarrow i$   
    RevDFS( $j$ )  
 $h \leftarrow h + 1$   
 $S[h] \leftarrow i$ 
```

RevDFS is a recursive procedure, identical to DFS, but with the reversed arcs.



Procedure *FindArc*(S)

/ Find a minimum reduced cost arc entering S */*

for $i \in \mathcal{N}$ **do**

$Flag[i] \leftarrow false$

for $u = 1, \dots, h$ **do**

$Flag[S[u]] \leftarrow true$

$\alpha \leftarrow \infty$

for $u = 1, \dots, h$ **do**

for $(i, S[u]) \in \delta^-(S[u])$ **do**

if $(\neg Flag[i]) \wedge (\bar{c}_{iS[u]} < \alpha)$ **then**

$\alpha \leftarrow \bar{c}_{iS[u]}$



Procedure *UpdateCost*(S, α)

/ Update the reduced costs */*

for $u = 1, \dots, h$ **do**

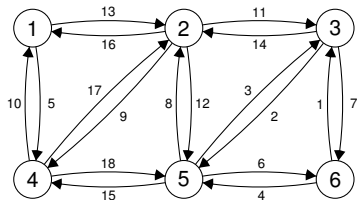
for $(i, S[u]) \in \delta^-(S[u])$ **do**

if $(\neg \text{Flag}[i])$ **then**

$\bar{c}_{iS[u]} \leftarrow \bar{c}_{iS[u]} - \alpha$



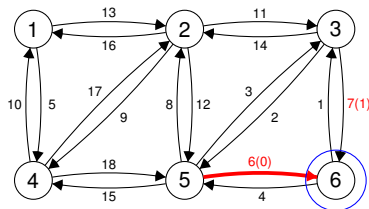
Example



Node 1 is the root.



Iteration 1



DFS : 1 2 3 4 5 6

Order : 1 2 3 4 5 6

$S = \{6\}$

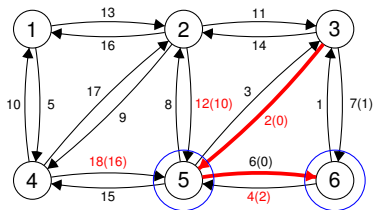
$\delta^-(S) = \{(3, 6), (5, 6)\}$

$\alpha = 6$

$$y_6 = 6$$



Iteration 2



DFS : 1 2 3 4 5 6

Order : 1 2 3 4 6 5

$S = \{5\}$

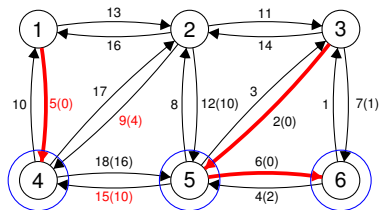
$\delta^-(S) = \{(2, 5), (3, 5), (4, 5), (6, 5)\}$

$\alpha = 2$

$$y_6 = 6 \quad y_5 = 2$$



Iteration 3



DFS : 1 2 3 5 6 4

Order : 1 2 6 5 3 4

$S = \{4\}$

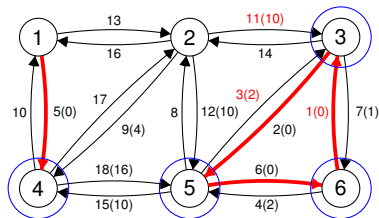
$\delta^-(S) = \{(1, 4), (2, 4), (5, 4)\}$

$\alpha = 5$

$$y_6 = 6 \quad y_5 = 2 \quad y_4 = 5$$



Iteration 4



DFS : 1 4 2 3 5 6

Order : 4 1 2 6 5 3

$S = \{3\}$

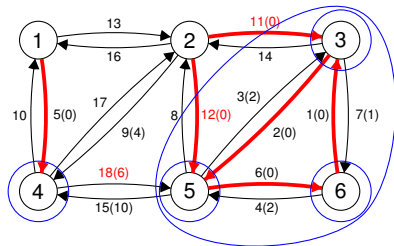
$\delta^-(S) = \{(2, 3), (5, 3), (6, 3)\}$

$\alpha = 1$

$$y_6 = 6 \quad y_5 = 2 \quad y_4 = 5 \quad y_3 = 1$$



Iteration 5



DFS : 1 4 2 3 5 6

Order : 4 1 2 6 5 3

$S = \{3, 5, 6\}$

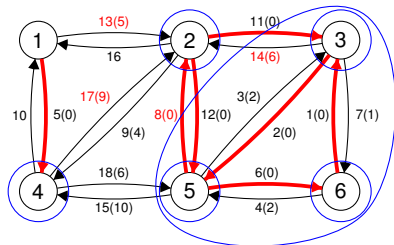
$\delta^-(S) = \{(2, 3), (2, 5), (4, 5)\}$

$\alpha = 10$

$$y_6 = 6 \quad y_5 = 2 \quad y_4 = 5 \quad y_3 = 1 \quad y_{356} = 10$$



Iteration 6



DFS : 1 4 2 3 5 6

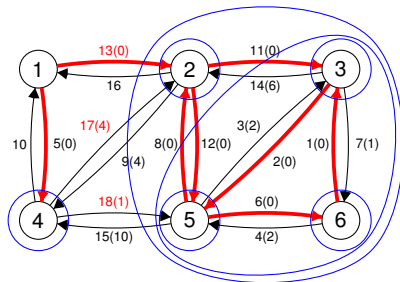
Order : 4 1 6 5 3 2

 $S = \{2\}$ $\delta^-(S) = \{(1, 2), (3, 2), (4, 2), (5, 2)\}$ $\alpha = 8$

$$y_6 = 6 \quad y_5 = 2 \quad y_4 = 5 \quad y_3 = 1 \quad y_{356} = 10 \quad y_2 = 8$$



Iteration 7



DFS : 1 4 2 3 5 6

Order : 4 1 6 5 3 2

$S = \{2, 3, 5, 6\}$

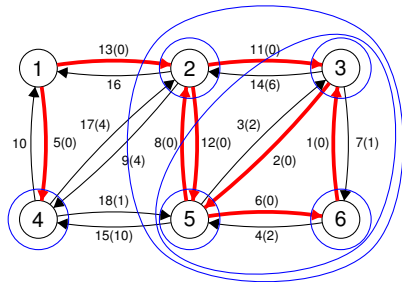
$\delta^-(S) = \{(1, 2), (4, 2), (4, 5)\}$

$\alpha = 5$

$$y_6 = 6 \quad y_5 = 2 \quad y_4 = 5 \quad y_3 = 1 \quad y_{356} = 10 \quad y_2 = 8 \quad y_{2356} = 5$$



Iteration 8



DFS : 1 4 2 3 5 6

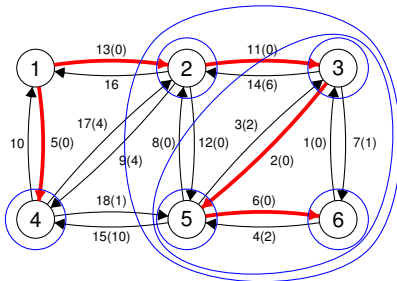
Order : 4 6 5 3 2 1

Node 1 is the root: STOP!

$$y_6 = 6 \quad y_5 = 2 \quad y_4 = 5 \quad y_3 = 1 \quad y_{356} = 10 \quad y_2 = 8 \quad y_{2356} = 5$$



Optimal solution



$$y_6 = 6 \quad y_5 = 2 \quad y_4 = 5 \quad y_3 = 1 \quad y_{356} = 10 \quad y_2 = 8 \quad y_{2356} = 5$$

$$w = \sum_S y_S = \mathbf{37}$$



Implementation 2

In this second approach, Edmonds algorithm is implemented using both primal and **dual data-structures**.

The implementation does not make use of **graph search sub-routines** and assumes that the digraph is **strongly connected**.

Step1 : DualAscent

Step2 : ComputeSolution(r)

- Step 1: build a sequence of **nested SCCs** in the digraph of admissible arcs, until a unique SCC is found containing all the nodes. The information about which node is the root is ignored. This step can be implemented in $O(m \log n)$.
- Step 2: find an optimal solution given a root r and the information stored in the **dual data-structure** built in Step 1. This step can be implemented in $O(n)$.



Step 1: dual ascent

SCCs with no **basic entering arcs** are iteratively found, so that they form a **path**. Each node in the path is a **basic SCC**.

A minimum cost arc $(u, v) \in \mathcal{A}$ is found among those entering the first node in the **path**, i.e. a certain **SCC S** ; the dual variable y_S is updated accordingly (**dual ascent**).

If u belongs to a SCC S' already in the **path**, then all **SCCs** in the **path** between S and S' are merged into a **single SCC**. Otherwise, a new **SCC** including only node u is created and appended at the beginning of the **path**.

Besides concatenating **basic SCCs** in a **path**, the algorithm also records how the **basic SCCs** are nested in one another forming a **tree** of **basic SCCs**.



Step 1: data structures

For every SCC S made of a single node, we define a priority queue $P(S)$ with the arcs entering it. When two SCCs S' and S'' are merged into a larger one, S , the corresponding priority queues $P(S')$ and $P(S'')$ are merged into a single priority queue $P(S)$.

A union-find data-structure *Comp* is used to find the SCCs to which specific nodes belong.

Data structures:

A vector *Arc*: the (unique) basic entering arc for each basic SCC.

A vector y : the values of the basic dual variables.

A vector *Next*: the next SCC for each SCC in the path.

A vector *Parent*: the parent SCC for each SCC in the tree.

A list *Children*: information symmetric to that of *Parent*.

All these vectors have maximum size $2n$.



Step 1: pseudo-code

InitializeStars

$k \leftarrow 0$

NewComponent(Random($\mathcal{N}$))

$s \leftarrow 1$

while $P(s) \neq \emptyset$ **do**

repeat

$(u, v) \leftarrow \text{ExtractMin}(P(s))$

until $\text{Comp}[u] \neq s$

$\text{Arc}[s] \leftarrow (u, v)$

$y[s] \leftarrow c(u, v)$

$\text{ReduceCosts}(P(s), y[s])$

if $\text{Comp}[u] = \text{nil}$ **then**

$\text{NewComponent}(u)$

$\text{Next}[k] \leftarrow s$

else

$\text{Merge}(\text{Comp}[u])$

$s \leftarrow k$



InitializeStars

```
for  $i \in \mathcal{N}$  do  
     $P(i) \leftarrow nil$   
     $Comp[i] \leftarrow nil$   
for  $(u, v) \in \mathcal{A}$  do  
     $InsertStar(u, c(u, v), \delta(v))$ 
```



CreateNewSCC

$$k \leftarrow k + 1$$

$$\text{Arc}[k] \leftarrow \text{nil}$$

$$y[k] \leftarrow 0$$

$$\text{Parent}[k] \leftarrow \text{nil}$$

$$\text{Children}[k] \leftarrow \text{nil}$$



NewComponent(u)

CreateNewSCC

$Comp[u] \leftarrow k$

$Leaf[u] \leftarrow k$

$P[k] \leftarrow \delta[u]$



Merge(t)

CreateNewSCC

$P[k] \leftarrow nil$

$Next[k] \leftarrow Next[t]$

$Stop \leftarrow false$

repeat

$Parent[s] \leftarrow k$

$Insert(s, Children(k))$

$P[k] \leftarrow MergePQ(P[s], P[k])$

$UpdateComp(s, k)$

if $s = t$ **then**

$Stop \leftarrow true$

else

$s \leftarrow Next[s]$

until $Stop$



Computational complexity of Step 1

The number of calls to *InsertStar* is $O(m)$ (all arcs are inserted in one PQ).

The number of calls to *ExtractMin* is $O(m)$ (each arc can be extracted at most once).

The number of calls to *MergePQ* is $O(n)$ (at most $2n$ SCCs become basic).

The number of calls to *ReduceCosts* is $O(n)$ (see above).

If all operations are implemented to take $O(\log n)$, then the overall complexity is $O(m \log n)$.

UpdateComp requires a Union-Find data-structure.



Step 2

While Step 1 is independent of r , Step 2 produces different solutions depending on r .

 $T \leftarrow \emptyset$ $RootList \leftarrow \emptyset$ $Dismantle(r)$ **while** $RootList \neq \emptyset$ **do** $s \leftarrow Extract(RootList)$ $(u, v) \leftarrow Arc[s]$ $T \leftarrow T \cup \{(u, v)\}$ $Dismantle(v)$ **return** T

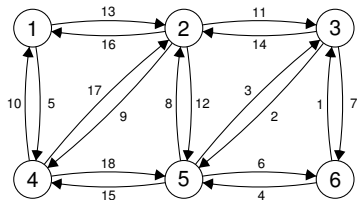


Dismantle(v)

```
s ← Leaf[v]  
while Parent[s] ≠ nil do  
  for t ∈ Children[Parent[s]] : t ≠ s do  
    Parent[t] ← nil  
    Insert(t, RootList)  
  s ← Parent[s]
```



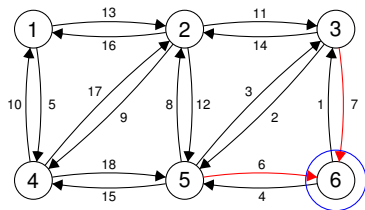
Example of Step 1



Assume node 6 is selected at random to start.



Dual iteration 1



Node	1	2	3	4	5	6
Comp						1
Leaf						1

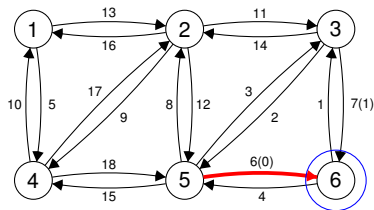
$$k = 1 \quad s = 1$$

$$P(1) = \{((5, 6), 6), ((3, 6), 7)\}$$

k	1
Arc	nil
y	0
Next	nil
Parent	nil
Children	nil



Primal iteration 1



Node	1	2	3	4	5	6
Comp						1
Leaf						1

$$k = 1 \quad s = 1$$

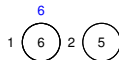
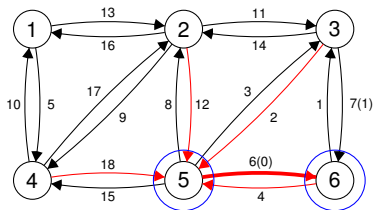
$$(u, v) = (5, 6) \quad \bar{c}_{56} = 6$$

$$P(1) = \{((3, 6), 1)\}$$

k	1
Arc	(5,6)
y	6
Next	nil
Parent	nil
Children	nil



Dual iteration 2



Node	1	2	3	4	5	6
Comp					2	1
Leaf					2	1

$k = 2 \quad s = 1$

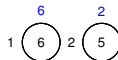
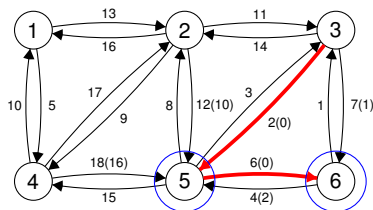
$$P(2) =$$

$$\{((3, 5), 2), ((6, 5), 4), ((2, 5), 12), ((4, 5), 18)\}$$

k	1	2
Arc	(5,6)	nil
y	6	0
Next	nil	1
Parent	nil	nil
Children	nil	nil



Primal iteration 2



Node	1	2	3	4	5	6
Comp					2	1
Leaf					2	1

$$k = 2 \quad s = 2$$

$$(u, v) = (3, 5) \quad \bar{c}_{35} = 2$$

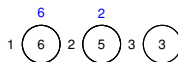
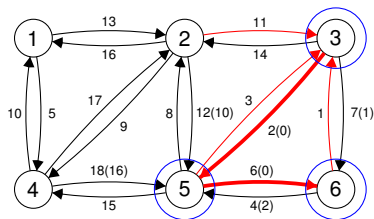
$$P(2) =$$

$$\{((6, 5), 2), ((2, 5), 10), ((4, 5), 16)\}$$

k	1	2
Arc	(5,6)	(3,5)
y	6	2
Next	nil	1
Parent	nil	nil
Children	nil	nil



Dual iteration 3



Node	1	2	3	4	5	6
Comp			3		2	1
Leaf			3		2	1

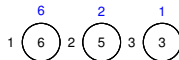
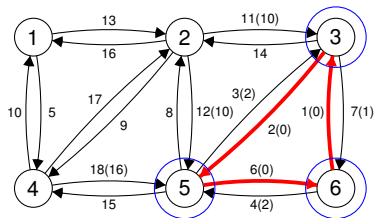
$$k = 3 \quad s = 2$$

$$P(3) = \{((6, 3), 1), ((5, 3), 3), ((2, 3), 11)\}$$

k	1	2	3
Arc	(5,6)	(3,5)	nil
y	6	2	0
Next	nil	1	nil
Parent	nil	nil	nil
Children	nil	nil	nil



Primal iteration 3



Node	1	2	3	4	5	6
Comp			3		2	1
Leaf			3		2	1

$$k = 3 \quad s = 3$$

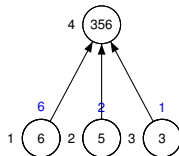
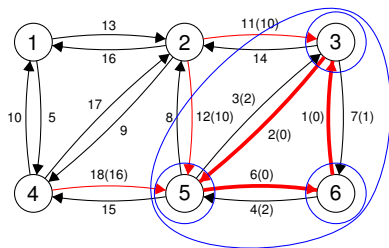
$$(u, v) = (6, 3) \quad \bar{c}_{63} = 1$$

$$P(3) = \{((5, 3), 2), ((2, 3), 10)\}$$

k	1	2	3
Arc	(5,6)	(3,5)	(6,3)
y	6	2	1
Next	nil	1	nil
Parent	nil	nil	nil
Children	nil	nil	nil



Dual iteration 4



Node	1	2	3	4	5	6
Comp			4		4	4
Leaf			3		2	1

$k = 4 \quad s = 3$

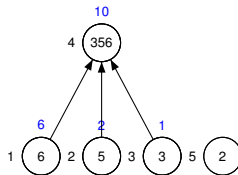
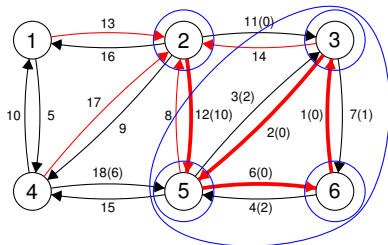
Merge $t = 1$

$P(4) = \{((3, 6), 1), ((6, 5), 2), ((5, 3), 2), ((2, 5), 10), ((2, 3), 10), ((4, 5), 16)\}.$

k	1	2	3	4
Arc	(5,6)	(3,5)	(6,3)	nil
y	6	2	1	0
Next	nil	1	nil	nil
Parent	4	4	4	nil
Children	nil	nil	nil	(1,2,3)



Dual iteration 5



Node	1	2	3	4	5	6
Comp		5	4		4	4
Leaf		5	3		2	1

$$k = 5 \quad s = 4$$

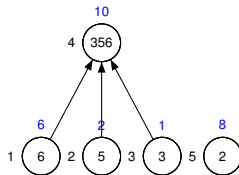
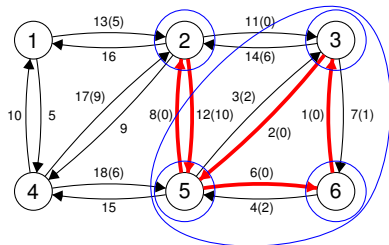
$$P(5) =$$

$$\{((5, 2), 8), ((1, 2), 13), ((3, 2), 14), ((4, 2), 17)\}.$$

k	1	2	3	4	5
Arc	(5,6)	(3,5)	(6,3)	(2,5)	nil
y	6	2	1	10	0
Next	nil	1	nil	nil	4
Parent	4	4	4	nil	nil
Children	nil	nil	nil	(1,2,3)	nil



Primal iteration 5



Node	1	2	3	4	5	6
Comp		5	4		4	4
Leaf		5	3		2	1

$$k = 5 \quad s = 5$$

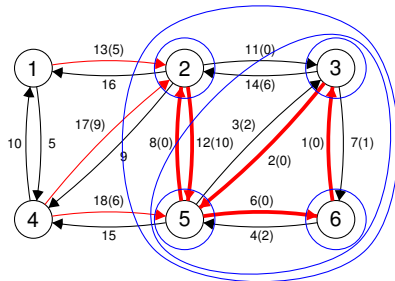
$$(u, v) = (5, 2) \quad \bar{c}_{52} = 8$$

$$P(5) = \{((1, 2), 5), ((3, 2), 6), ((4, 2), 9)\}$$

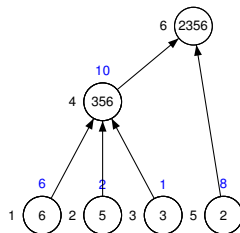
k	1	2	3	4	5
Arc	(5,6)	(3,5)	(6,3)	(2,5)	(5,2)
y	6	2	1	10	8
Next	nil	1	nil	nil	4
Parent	4	4	4	nil	nil
Children	nil	nil	nil	(1,2,3)	nil



Dual iteration 6



Node	1	2	3	4	5	6
Comp		6	6		6	6
Leaf		5	3		2	1



$k = 6 \quad s = 5$

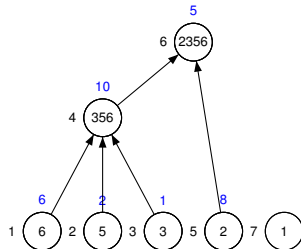
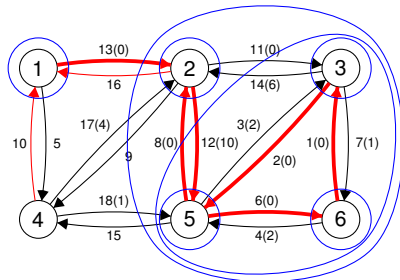
Merge $t = 4$.

$P(6) = \{((2, 3), 0), ((1, 2), 5), ((4, 5), 6), ((3, 2), 6), ((4, 2), 9)\}$

k	1	2	3	4	5	6
Arc	(5,6)	(3,5)	(6,3)	(2,5)	(5,2)	nil
y	6	2	1	10	8	0
Next	nil	1	nil	nil	4	nil
Parent	4	4	4	6	6	nil
Children	nil	nil	nil	(1,2,3)	nil	(4,5)



Dual iteration 7



Node	1	2	3	4	5	6
Comp	7	6	6		6	6
Leaf	7	5	3		2	1

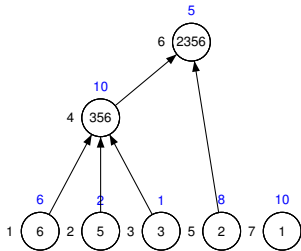
$$k = 7 \quad s = 6$$

$$P(7) = \{((4, 1), 10), ((2, 1), 16)\}$$

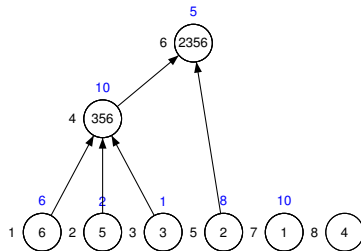
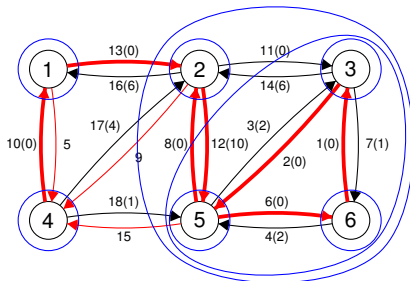
k	1	2	3	4	5	6	7
Arc	(5,6)	(3,5)	(6,3)	(2,5)	(5,2)	(1,2)	nil
y	6	2	1	10	8	5	0
Next	nil	1	nil	nil	4	nil	6
Parent	4	4	4	6	6	nil	nil
Children	nil	nil	nil	(1,2,3)	nil	(4,5)	nil



Primal iteration 7


$$k = 7 \quad s = 7$$
$$(u, v) = (4, 1) \quad \bar{c}_{41} = 10$$
$$P(7) = \{((2, 1), 6)\}$$

Dual iteration 8



Node	1	2	3	4	5	6
Comp	7	6	6	8	6	6
Leaf	7	5	3	8	2	1

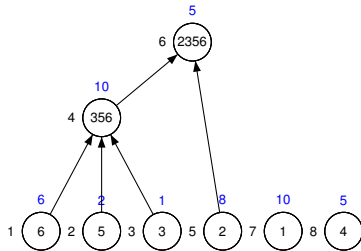
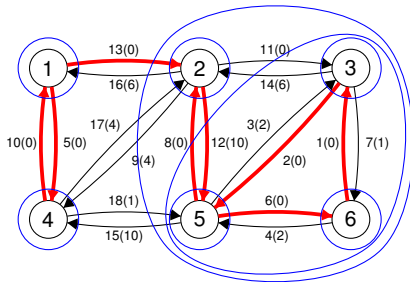
$$k = 8 \quad s = 7$$

$$P(8) = \{((1, 4), 5), ((2, 4), 9), ((5, 4), 15)\}$$

k	1	2	3	4	5	6	7	8
Arc	(5,6)	(3,5)	(6,3)	(2,5)	(5,2)	(1,2)	(4,1)	nil
y	6	2	1	10	8	5	10	0
Next	nil	1	nil	nil	4	nil	6	7
Parent	4	4	4	6	6	nil	nil	nil
Children	nil	nil	nil	(1,2,3)	nil	(4,5)	nil	nil



Primal iteration 8



Node	1	2	3	4	5	6
Comp	7	6	6	8	6	6
Leaf	7	5	3	8	2	1

$$k = 8 \quad s = 8$$

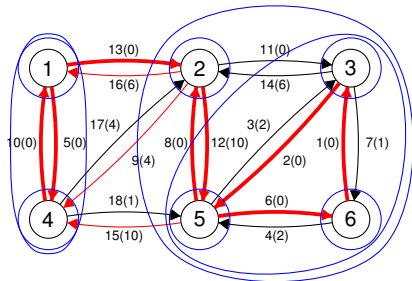
$$(u, v) = (1, 4) \quad \bar{c}_{14} = 5$$

$$P(8) = \{((2, 4), 4), ((5, 4), 10)\}$$

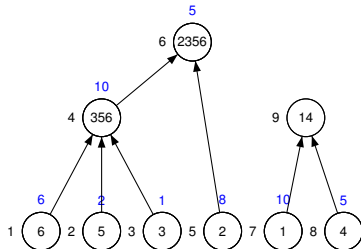
k	1	2	3	4	5	6	7	8
Arc	(5,6)	(3,5)	(6,3)	(2,5)	(5,2)	(1,2)	(4,1)	(1,4)
y	6	2	1	10	8	5	10	5
Next	nil	1	nil	nil	4	nil	6	7
Parent	4	4	4	6	6	nil	nil	nil
Children	nil	nil	nil	(1,2,3)	nil	(4,5)	nil	nil



Dual iteration 9



Node	1	2	3	4	5	6
Comp	9	6	6	9	6	6
Leaf	7	5	3	8	2	1



$k = 9 \quad s = 8$

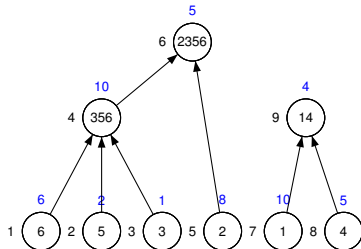
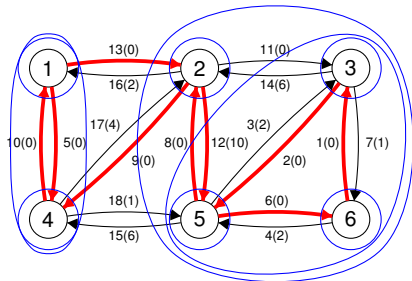
Merge $t = 7$.

$P(9) = \{((2, 4), 4), ((2, 1), 6), ((5, 4), 10)\}$

k	1	2	3	4	5	6	7	8	9
Arc	(5,6)	(3,5)	(6,3)	(2,5)	(5,2)	(1,2)	(4,1)	(1,4)	nil
y	6	2	1	10	8	5	10	5	0
Next	nil	1	nil	nil	4	nil	6	7	6
Parent	4	4	4	6	6	nil	9	9	nil
Children	nil	nil	nil	(1,2,3)	nil	(4,5)	nil	nil	(7,8)



Primal iteration 9



Node	1	2	3	4	5	6
Comp	9	6	6	9	6	6
Leaf	7	5	3	8	2	1

$$k = 9 \quad s = 9$$

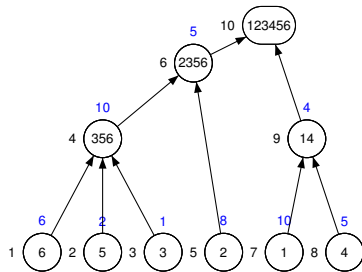
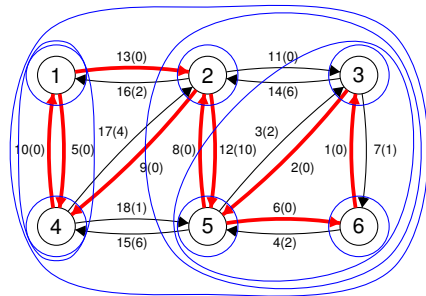
$$(u, v) = (2, 4) \quad \bar{c}_{24} = 4$$

$$P(9) = \{((2, 1), 2), ((5, 4), 6)\}$$

k	1	2	3	4	5	6	7	8	9
Arc	(5,6)	(3,5)	(6,3)	(2,5)	(5,2)	(1,2)	(4,1)	(1,4)	(2,4)
y	6	2	1	10	8	5	10	5	4
Next	nil	1	nil	nil	4	nil	6	7	6
Parent	4	4	4	6	6	nil	9	9	nil
Children	nil	nil	nil	(1,2,3)	nil	(4,5)	nil	nil	(7,8)



Dual iteration 10



Node	1	2	3	4	5	6
Comp	10	10	10	10	10	10
Leaf	7	5	3	8	2	1

$k = 10 \quad s = 9$

Merge $t = 6$.

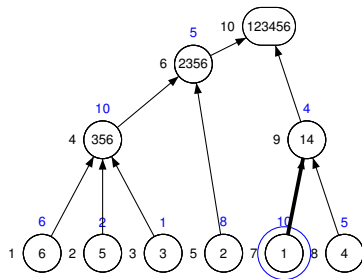
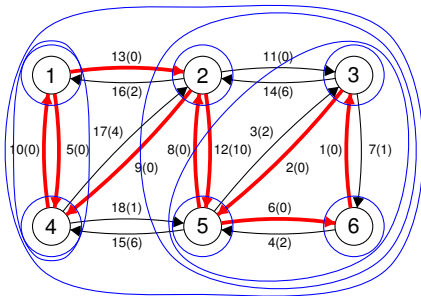
$P(10) =$

$\{((4, 5), 2), ((2, 1), 2), ((4, 2), 4), ((5, 4), 6)\}.$

k	1	2	3	4	5	6	7	8	9	10
Arc	(5,6)	(3,5)	(6,3)	(2,5)	(5,2)	(1,2)	(4,1)	(1,4)	(2,4)	nil
y	6	2	1	10	8	5	10	5	4	0
Next	nil	1	nil	nil	4	nil	6	7	6	nil
Parent	4	4	4	6	6	10	9	9	10	nil
Children	nil	nil	nil	(1,2,3)	nil	(4,5)	nil	nil	(7,8)	(6,9)



Example: $r = 1$



Node	1	2	3	4	5	6
Leaf	7	5	3	8	2	1

$RootList = \emptyset$

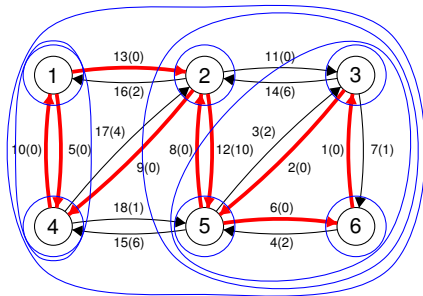
$s = Leaf(1) = 7, Parent(s) = 9$

$T = \emptyset$

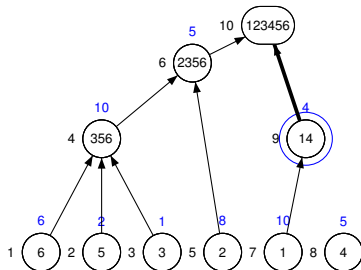
k	1	2	3	4	5	6	7	8	9	10
Arc	(5,6)	(3,5)	(6,3)	(2,5)	(5,2)	(1,2)	(4,1)	(1,4)	(2,4)	nil
y	6	2	1	10	8	5	10	5	4	0
Next	nil	1	nil	nil	4	nil	6	7	6	nil
Parent	4	4	4	6	6	10	9	9	10	nil
Children	nil	nil	nil	(1,2,3)	nil	(4,5)	nil	nil	(7,8)	(6,9)



Example: $r = 1$



Node	1	2	3	4	5	6
Leaf	7	5	3	8	2	1

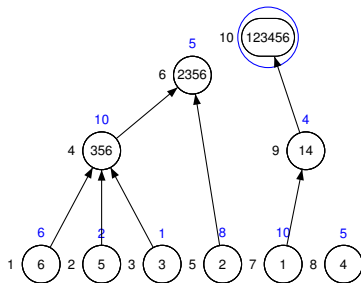
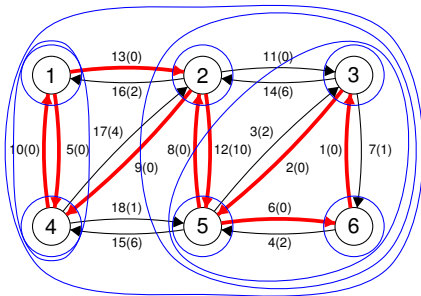


$RootList = \{8\}$
 $s = 9, Parent(s) = 10$
 $T = \emptyset$

k	1	2	3	4	5	6	7	8	9	10
Arc	(5,6)	(3,5)	(6,3)	(2,5)	(5,2)	(1,2)	(4,1)	(1,4)	(2,4)	nil
y	6	2	1	10	8	5	10	5	4	0
Next	nil	1	nil	nil	4	nil	6	7	6	nil
Parent	4	4	4	6	6	10	9	9	10	nil
Children	nil	nil	nil	(1,2,3)	nil	(4,5)	nil	nil	(7,8)	(6,9)



Example: $r = 1$



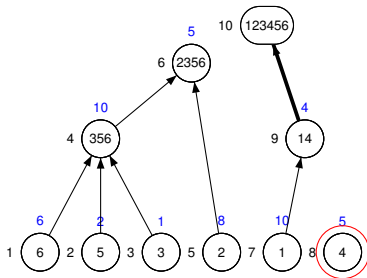
Node	1	2	3	4	5	6
Leaf	7	5	3	8	2	1

$RootList = \{8, 6\}$
 $s = 10, Parent(s) = nil$
 $T = \emptyset$

k	1	2	3	4	5	6	7	8	9	10
Arc	(5,6)	(3,5)	(6,3)	(2,5)	(5,2)	(1,2)	(4,1)	(1,4)	(2,4)	nil
y	6	2	1	10	8	5	10	5	4	0
Next	nil	1	nil	nil	4	nil	6	7	6	nil
Parent	4	4	4	6	6	10	9	9	10	nil
Children	nil	nil	nil	(1,2,3)	nil	(4,5)	nil	nil	(7,8)	(6,9)

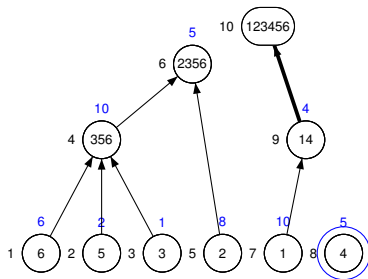


Example: $r = 1$


$$\begin{aligned} \text{RootList} &= \{6\} \\ s &= 8, \text{Arc}(s) = (1, 4), \text{Leaf}(4) = 8 \\ T &= \{(1, 4)\} \end{aligned}$$

nil
 (6,9) 
 UNIVERSITÀ DEGLI STUDI DI MILANO

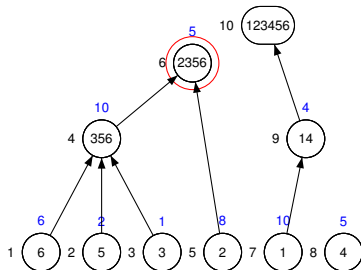
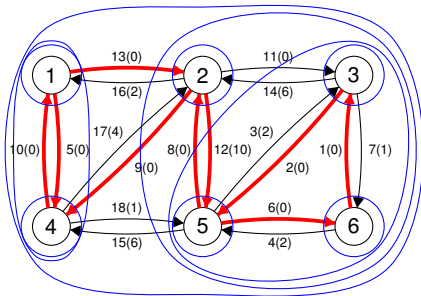
Example: $r = 1$


$$\begin{aligned} \text{RootList} &= \{6\} \\ s &= 8, \text{Parent}(s) = \text{nil} \\ T &= \{(1, 4)\} \end{aligned}$$

nil
(6,9)

UNIVERSITÀ DEGLI STUDI DI MILANO

Example: $r = 1$



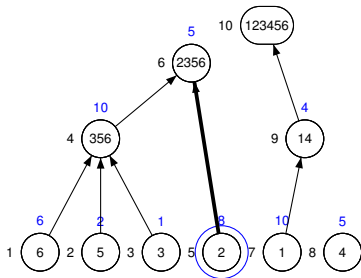
Node	1	2	3	4	5	6
Leaf	7	5	3	8	2	1

$RootList = \{\}$
 $s = 6, Arc(s) = (1, 2), Leaf(2) = 5$
 $T = \{(1, 4), (1, 2)\}$

k	1	2	3	4	5	6	7	8	9	10
Arc	(5,6)	(3,5)	(6,3)	(2,5)	(5,2)	(1,2)	(4,1)	(1,4)	(2,4)	nil
y	6	2	1	10	8	5	10	5	4	0
Next	nil	1	nil	nil	4	nil	6	7	6	nil
Parent	4	4	4	6	6	10	9	9	10	nil
Children	nil	nil	nil	(1,2,3)	nil	(4,5)	nil	nil	(7,8)	(6,9)

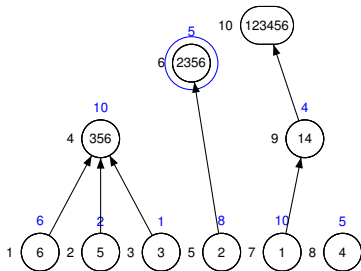


Example: $r = 1$


$$\begin{aligned} \text{RootList} &= \{\} \\ s &= 5, \text{Parent}(5) = 6 \\ T &= \{(1, 4), (1, 2)\} \end{aligned}$$

nil
 (6,9) 
 UNIVERSITÀ DEGLI STUDI DI MILANO

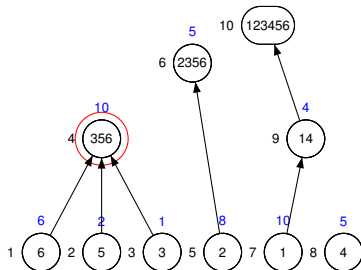
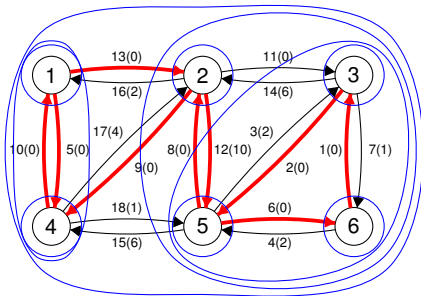
Example: $r = 1$


$$\begin{aligned} \text{RootList} &= \{4\} \\ s &= 6, \text{Parent}(6) = \text{nil} \\ T &= \{(1, 4), (1, 2)\} \end{aligned}$$

nil
(6,9)

UNIVERSITÀ DEGLI STUDI DI MILANO

Example: $r = 1$



Node	1	2	3	4	5	6
Leaf	7	5	3	8	2	1

$RootList = \{ \}$

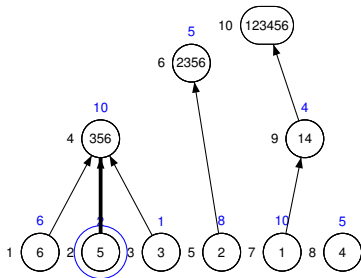
$s = 4, Arc(s) = (2, 5), Leaf(5) = 2$

$T = \{(1, 4), (1, 2), (2, 5)\}$

k	1	2	3	4	5	6	7	8	9	10
Arc	(5,6)	(3,5)	(6,3)	(2,5)	(5,2)	(1,2)	(4,1)	(1,4)	(2,4)	nil
y	6	2	1	10	8	5	10	5	4	0
Next	nil	1	nil	nil	4	nil	6	7	6	nil
Parent	4	4	4	6	6	10	9	9	10	nil
Children	nil	nil	nil	(1,2,3)	nil	(4,5)	nil	nil	(7,8)	(6,9)



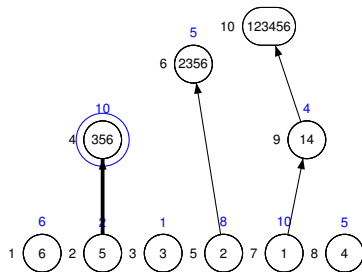
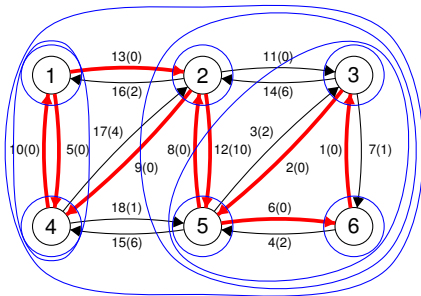
Example: $r = 1$


$$\begin{aligned} \text{RootList} &= \{\} \\ s &= 2, \text{Parent}(s) = 4 \\ T &= \{(1, 4), (1, 2), (2, 5)\} \end{aligned}$$

nil
(6,9)


UNIVERSITÀ DEGLI STUDI DI MILANO

Example: $r = 1$



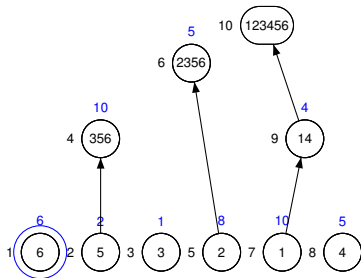
Node	1	2	3	4	5	6
Leaf	7	5	3	8	2	1

$RootList = \{1, 3\}$
 $s = 4, Parent(s) = nil$
 $T = \{(1, 4), (1, 2), (2, 5)\}$

k	1	2	3	4	5	6	7	8	9	10
Arc	(5,6)	(3,5)	(6,3)	(2,5)	(5,2)	(1,2)	(4,1)	(1,4)	(2,4)	nil
y	6	2	1	10	8	5	10	5	4	0
Next	nil	1	nil	nil	4	nil	6	7	6	nil
Parent	4	4	4	6	6	10	9	9	10	nil
Children	nil	nil	nil	(1,2,3)	nil	(4,5)	nil	nil	(7,8)	(6,9)



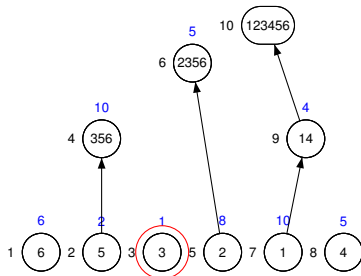
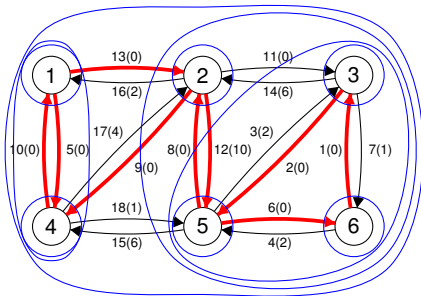
Example: $r = 1$


$$\begin{aligned} \text{RootList} &= \{3\} \\ s &= 1, \text{Parent}(s) = \text{nil} \\ T &= \{(1, 4), (1, 2), (2, 5), (5, 6)\} \end{aligned}$$

nil
(6,9)

UNIVERSITÀ DEGLI STUDI DI MILANO

Example: $r = 1$



Node	1	2	3	4	5	6
Leaf	7	5	3	8	2	1

$RootList = \{\}$

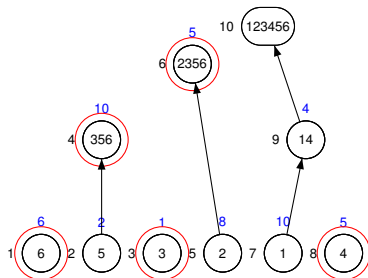
$s = 3$, $Arc(s) = (6, 3)$, $Leaf(3) = 3$

$T = \{(1, 4), (1, 2), (2, 5), (5, 6), (6, 3)\}$

k	1	2	3	4	5	6	7	8	9	10
Arc	(5,6)	(3,5)	(6,3)	(2,5)	(5,2)	(1,2)	(4,1)	(1,4)	(2,4)	nil
y	6	2	1	10	8	5	10	5	4	0
Next	nil	1	nil	nil	4	nil	6	7	6	nil
Parent	4	4	4	6	6	10	9	9	10	nil
Children	nil	nil	nil	(1,2,3)	nil	(4,5)	nil	nil	(7,8)	(6,9)



Example: $r = 1$


$$T^* = \{(1, 4), (1, 2), (2, 5), (5, 6), (6, 3)\}$$

$$z^* = 37$$

nil
(6,9)


UNIVERSITÀ DEGLI STUDI DI MILANO