

Maintain tree structure (T, L, U)

tree edges lower upper

- ① identify a cycle
↳ pivot cycle
 - ② compute and update
x flow and p dual values
 - ③ calculate θ and determine
the leaving arc
-

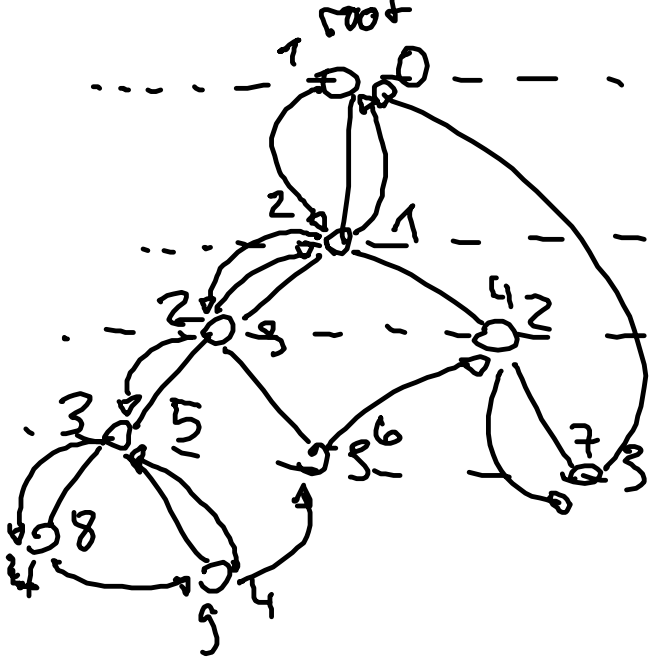
Saving the tree by indices at the nodes.

pred

depth

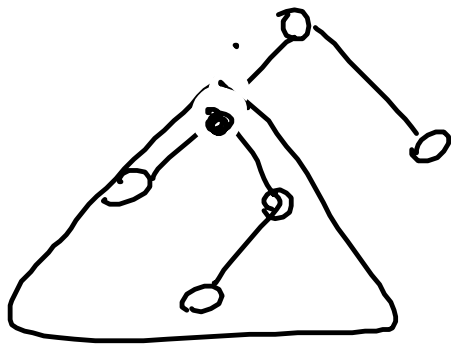
thead

,



depth first search

i	1	2	3	4	
$pred(i)$	-	1	2	2	
$depth(i)$	0	1	2	2	...
$thread(i)$	2	3	5	7	



We can visit all nodes of a subtree without visiting the others by following the thread index.

Given the tree structure compute the dual node potentials

```

begin
   $p(1) := 0;$ 
   $j := thread(1);$ 
  while  $j \neq 1$  do
  
```

```

begin
  i := pred(j);
  IF (i, j) ∈ T THEN
    p(j) := p(i) - cij;
    └── IF (j, i) ∈ T THEN
      p(j) := p(i) + cij;
  j := thread(j);
END;

```

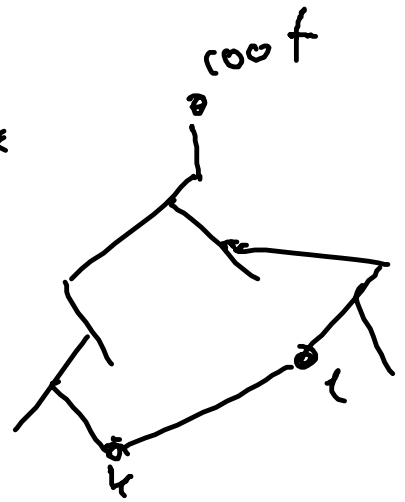
END;

Given the entering edge (k, l)
 We want to identify the pivot cycle

```

begin
  i := k;
  j := l;
  while i ≠ j do
    begin
      if depth(i) > depth(j) then
        i := pred(i);
      else if depth(j) > depth(i) then
        j := pred(j);
      else
        i := pred(i);
        j := pred(j);
      end;
    end;
  end;
end;

```

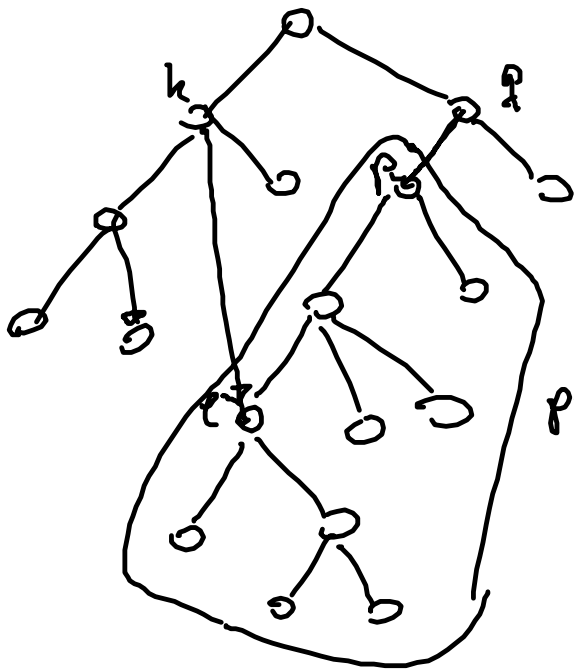


We use this method twice with two different side effects:

FIRST: Compute θ and leaving edge

SECOND: Update x flow values

FIRST update the node potentials



(p, q) leaving arc
 (k, l) entering arc
 $p(i) - p(j) = c_{ij} \quad \forall (i, j) \in T$

p -values will change

$$p(e)_{\text{new}} = p(k) - c_{ke}$$

$$\begin{aligned} p(e)_{\text{new}} - p(e)_{\text{old}} &= p(k) - c_{ke} - p(e)_{\text{old}} \\ &= -(c_{ke} - p(k) + p(e)_{\text{old}}) \\ &= -\bar{c}_{kl} \end{aligned}$$

IF l lies in the blue area, all p -value decrease by \bar{c}_{kl} .

IF k lies in the blue area, all p -value increase by \bar{c}_{kl} .

begin

$y := p;$

IF l lies in blue area THEN $change := -\bar{c}_{kl};$

IF k lies $\quad \quad \quad$ THEN $change := \bar{c}_{kj};$

$p(y) := p(y) + change;$

$y := thread(y);$

while $depth(y) > depth(p)$ do

begin

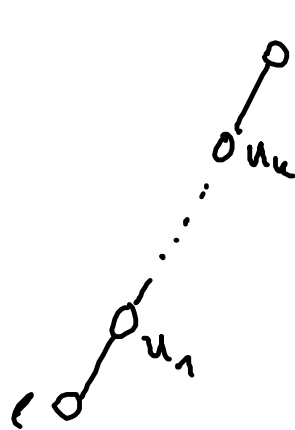
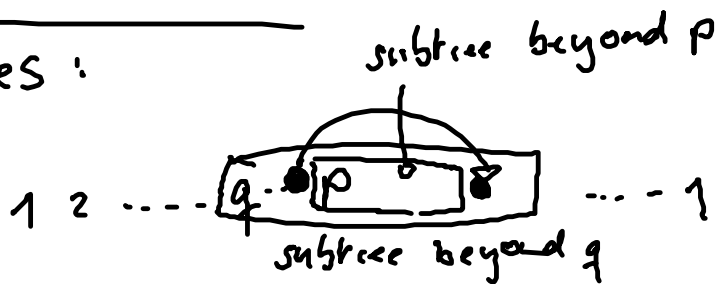
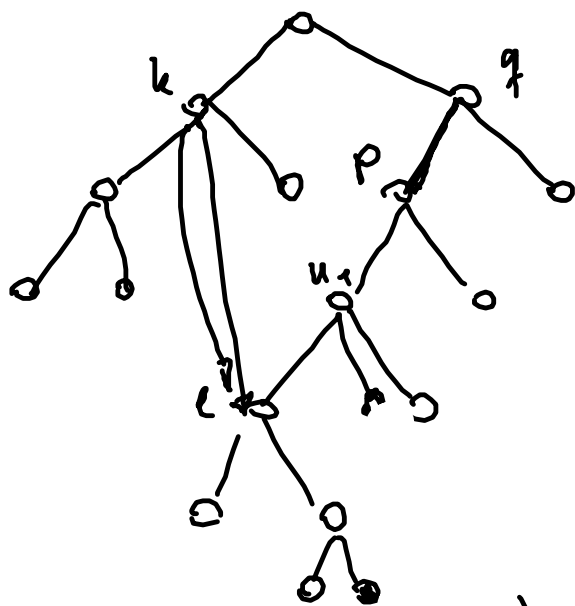
$p(y) := p(y) + change;$

$y := thread(y);$

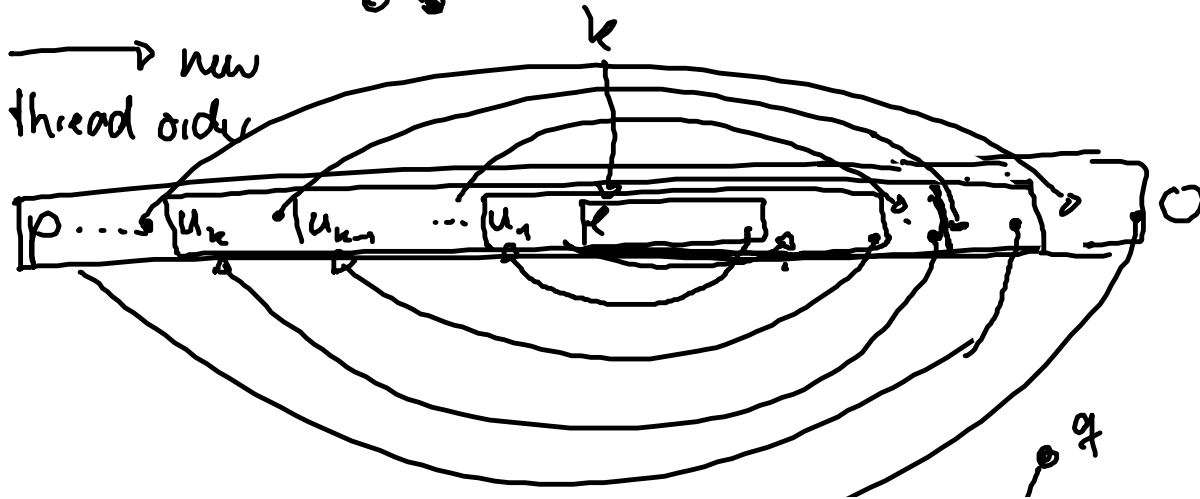
end;

end;

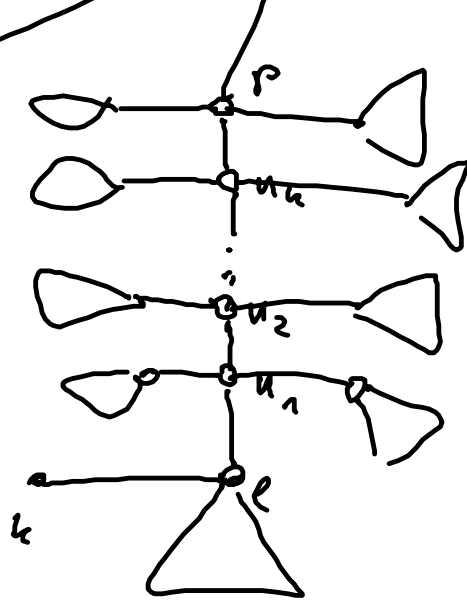
Update the thread indices:



u_1, \dots, u_n are the inner nodes of the l - p -Path in tree T .



$$x = \text{thread}(k)$$



- 1) store $k, e, p, \text{thread}(k)$
- 2) identify (by pred-indices) and store u_1, \dots, u_k
- 3) Walk over subtree beyond p using the old thread-values and update them

thread-indices are updated correctly.

pred-indices are easy to update:

One has just to invert the $e-p$ -path



Do this as a side effect when identifying u_1, \dots, u_k .

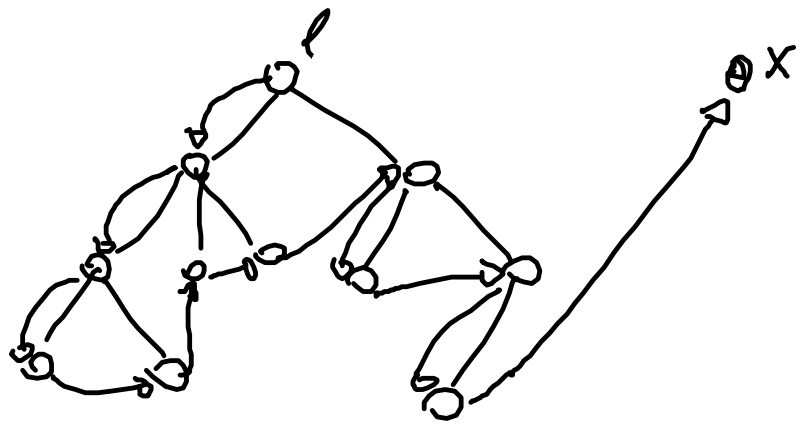
pred-indices are updated correctly

depth-indices:

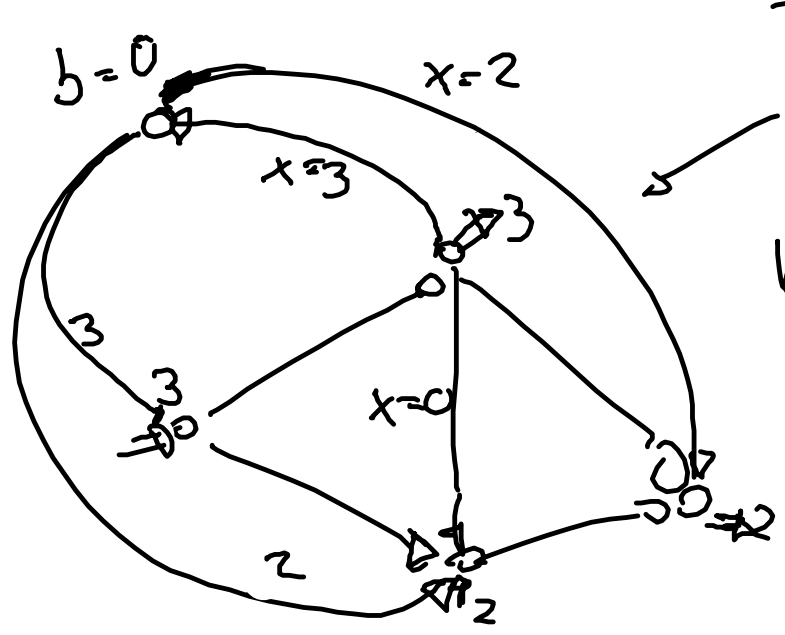
```

begin
  y := l;
  depth(y) := depth(l) + 1;
  y := thread(y);
  while y ≠ x do
  begin
    depth(y) := depth(pred(y)) + 1;
    y := thread(y);
  end;
end;

```



Depth-indices are updated correctly.



$\text{upper capacities} = \infty (=M)$
 $\text{cost} = \infty (=M = \sum_{c_{ij} > 0} c_{ij} + 1)$

Apply network simplex .

If one artificial arc is used in optimal solution the original problem is infeasible.