1

a) liczba wierzchołków w T

```
count (v):
  if v == NULL
    return 0;
  else
    return ( count ( left[v] ) + count ( right[v] ) + 1 )
```

b) maksymalna odległość między wierzchołkami w T

```
longest (v):
  if v == NULL
    return { 0, -1 }
  else
    { max_l , up_l } = longest ( left [v] )
    { max_r , up_r } = longest ( right [v] )

    return { max ( max_l , max_r , up_l + up_r + 2 ), max ( up_l, up_r ) + 1 }
```

3 Q - zbiór wierzchołków o stopniu wchodzącym 0

Wstaw wierzchołki o stopniu wchodzącym 0 do Q

dopóki Q niepuste

    wypisz i usuń najmniejszy element n z Q

    dla każdego wierzchołka m o krawędzi e z n do m:

        usuń e z grafu

        jeżeli m ma stopień wejściowy 0

            dodaj m do Q

3 Graf - lista sąsiedztwa

Q - kolejka priorytetowa (zaimplementowana na kopcu o min w korzeniu), trzymamy w niej wierzchołki o stopniu wchodzącym 0

dopóki Q niepusta:

  usuń n z przodu kolejki Q

  wypisz n

  dla każdego wierzchołka m o krawędzi e z n do m:

    usuń e z grafu

    jeżeli m ma stopień wejściowy 0

      wstaw m do Q

Alg: $\underbrace{\_\ \_\ \_ \ldots \_}_{=}$ y _ .. x .. _

Opt: $\overbrace{\_\ \_\ \_ \ldots \_}$ x _ .. y .. _

1° $y < x$

2° $y > x$

$$u \to u_1 \to \dots \to u_{k-1} \to u_k \to \dots \to v$$

Istnieje droga z u_k do v krótsza od każdej drogi z u_{k-1} do v

$$\Updownarrow$$

Najkrótsza droga z u_k do v jest krótsza od najkrótszej drogi z u_{k-1} do v

[redacted]

Dijkstra (v) znajduje najkrótsze drogi z v do każdego wierzchołka, czyli z każdego wierzchołka do v

Sortujemy wierzchołki od najdalszego do najbliższego

$$d[v_i] \geq \dots \geq d[u] \geq \dots \geq d[v_j] \geq d[u_k] \geq d[v]$$

Usuwamy krawędzie między wierzchołkami o tym samym dystansie
Zaczynając od prawej zliczamy sensowne drogi dla każdego wierzchołka do u.

[redacted]

Zliczamy sumując liczbę sensownych dróg z połączonych wierzchołków po prawej stronie.

Zwracamy policzoną wartość dla u

5

Sortujemy graf topologicznie

Liczymy maksymalne długości drogi od prawej:

$max\_len[v] = max(max\_len[u_1]+1, \ldots, max\_len[u_i]+1, 0)$ u_k - sąsiedzi v $k \in [1; i]$

wypisz max z max_len[]

Odzyskanie drogi

Sortujemy graf topologicznie

Liczymy maksymalne długości drogi od prawej

max_len[v] = 0

nast [v] = NULL

Dla każdego sąsiada v (u)

    Jeżeli max_len[u]+1 > max_len[v]

        max_len[v] = max_len[u]+1

        nast[v] = u

Znajdujemy v t.ż. max z max_len[] = max_len[v]

Dopóki v != NULL

    wypisz v

    v = nast[v]

6 Posortowana tablica a n-elementowa

left - pozycja pierwszego elementu w tablicy

right - pozycja $\lfloor \frac{n}{2} \rfloor + 1$ elementu w tablicy

dopóki left < $\lfloor \frac{n}{2} \rfloor + 1$ i right < n :

    jeżeli (2 · a[left] <= a[right]

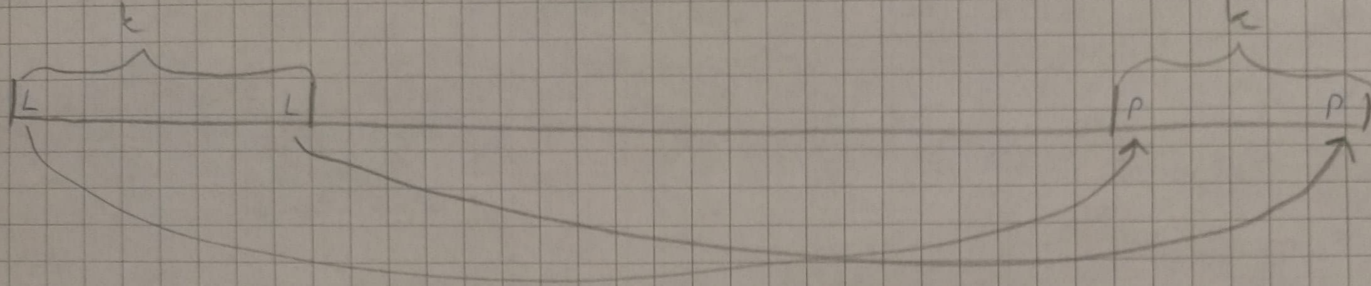
        count = count + 2

        left ++

    right ++

return count

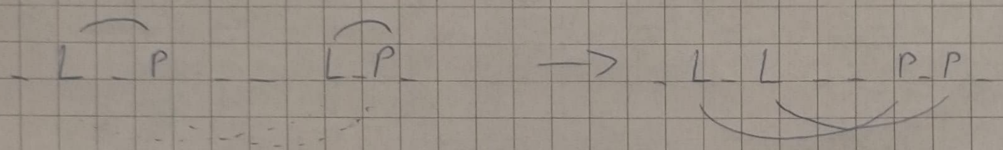
Pokażemy że rozwiązanie może być postaci



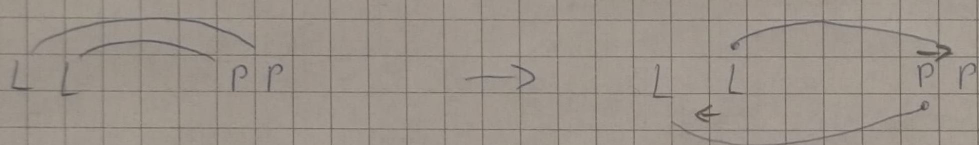
Element k-i-ty usuwamy w parze z n-i-tym

Weźmy rozwiązanie optymalne, które będziemy przekształcać

1°



2°



3°