AOD Lista 4

Kamil Włodarski

25 czerwca 2023

1 Wstęp

W sprawozdaniu opisane są implementacje i złożoność czasowa algorytmów z listy. Zostały one również porównane na podstawie wygeneryowanych danych testowych.

2 Algorytm Edmondsa-Karpa

2.1 Opis

Algorytm Edmondsa-Karpa jest specjalnym przypadkiem algorytmu Forda-Fulkersona. Znajduje maksymalny przepływ w grafie poprzez wielokrotne wyszukiwanie ścieżek powiększających w grafie.

2.2 Implementacja

Implementacja algorytmu Edmondsa-Karpa może być opisana następującym pseudokodem:

```
function EdmondsKarp(G, s, t) is
  for each edge (u, v) in G.E do
    (u, v).capacity = 0
    (u, v).flow = 0

  while exists a path p from s to t in the residual network do
    let cf be the capacity of the residual network Gf on p
    for each edge (u, v) in p do
      if (u, v) is a forward edge then
        (u, v).flow = (u, v).flow + cf
      else
        (u, v).flow = (u, v).flow − cf
  return the maximum flow in the network
```

Powyższy pseudokod opisuje podstawowy schemat algorytmu. Najważniejszym elementem algorytmu jest wyznaczanie ścieżki powiększającej, co odbywa się w każdej iteracji algorytmu.

2.3 Złożoność

Złożoność obliczeniowa algorytmu Edmondsa-Karpa to $O(VE^2)$, gdzie V to liczba wierzchołków, a E to liczba krawędzi w grafie.

3 Algorytm Dinica

3.1 Opis

Algorytm Dinica, podobnie jak algorytm Edmondsa-Karpa, jest realizacją idei Forda-Fulkersona. Algorytm Dinica wykorzystuje jednak strukturę sieci warstwowej (layered network), co pozwala na efektywniejsze wyszukiwanie ścieżek powiększających.

3.2 Implementacja

Implementacja algorytmu Dinica może być opisana następującym pseudokodem:

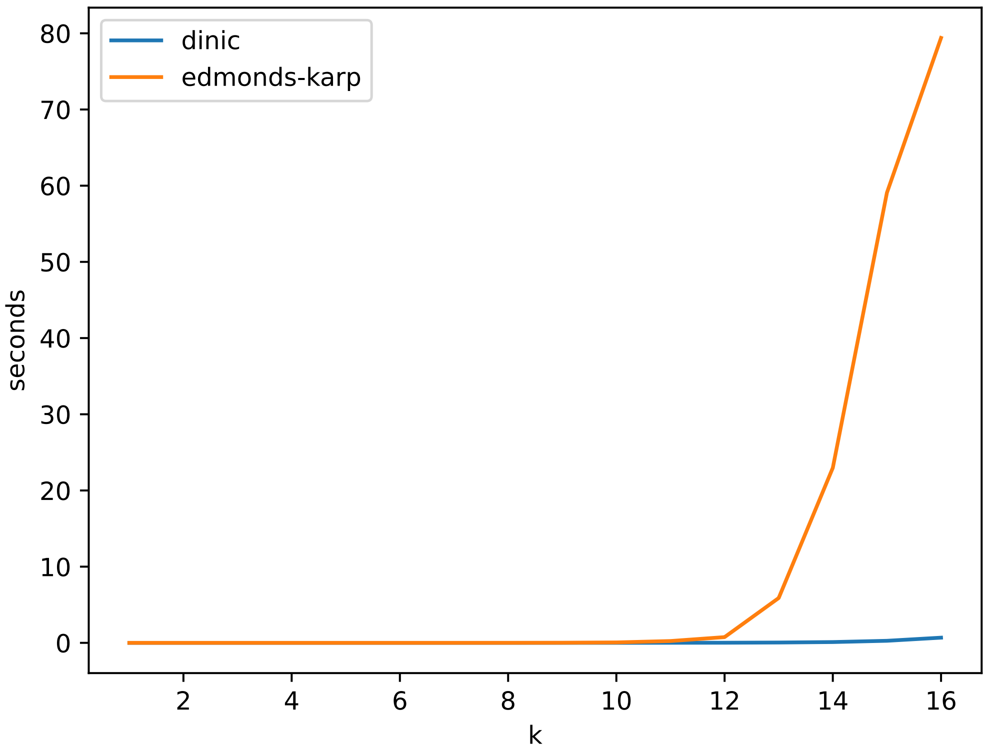
```
function Dinic(G, s, t) is
  while there exists a blocking flow B in the residual network do
    find the maximum flow f in B
    for each edge (u, v) in B do
      if (u, v) is a forward edge then
        (u, v).flow = (u, v).flow + f
      else
        (u, v).flow = (u, v).flow - f
  return the maximum flow in the network
```

Powyższy pseudokod ilustruje podstawową strukturę algorytmu Dinica. Kluczowym krokiem jest wyznaczanie blokującego przepływu w sieci warstwowej, co jest unikalne dla tego algorytmu.

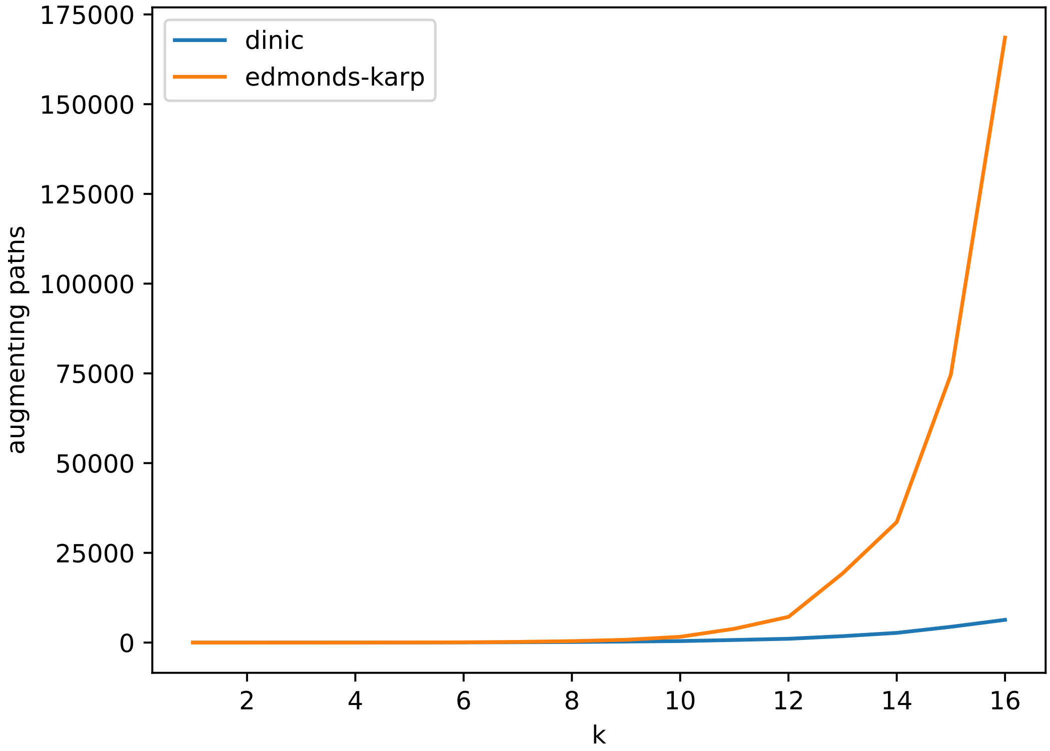
3.3 Złożoność

Złożoność obliczeniowa algorytmu Dinica to $O(V^2E)$ w najgorszym przypadku.

3.4 Porównanie



Rysunek 1: Porówanie czasu wykonywanie w zależności od k



Rysunek 2: Porównanie ilości iteracji algorytmu w zależności od k

4 Zadanie 2

Zadanie drugie polegało na opracowaniu algorytmu rozwiązującego problem maksymalnego skojarzenia w grafie dwudzielnym. Napisany prze zemnie algorytm bazuje na algorytmie Hopcrofta-Karpa.

4.1 implementacja

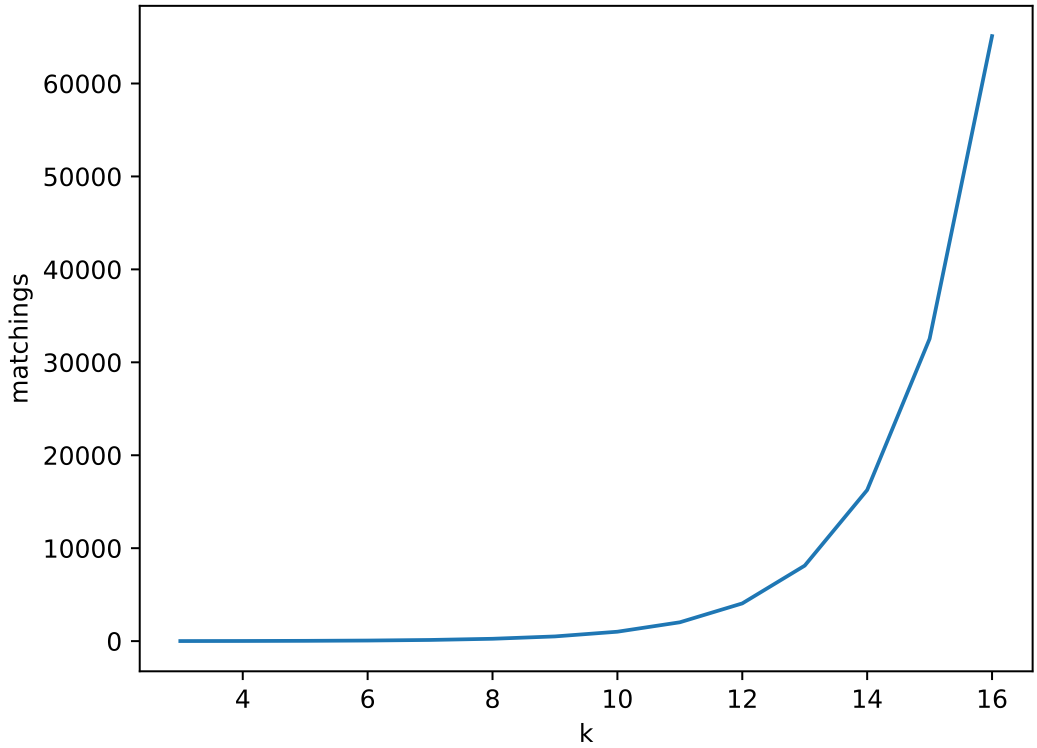
```
function BFS() is
    for each u in U do
        if Pair_U[u] = NIL then
            Dist[u] := 0
            Enqueue(Q, u)
        else
            Dist[u] := inf
    Dist[NIL] := inf
    while Empty(Q) = false do
        u := Dequeue(Q)
        if Dist[u] < Dist[NIL] then
```

```
            for each v in Adj[u] do
                if Dist[Pair_V[v]] = inf then
                    Dist[Pair_V[v]] := Dist[u] + 1
                    Enqueue(Q, Pair_V[v])
    return Dist[NIL] != inf

function DFS(u) is
    if u != NIL then
        for each v in Adj[u] do
            if Dist[Pair_V[v]] = Dist[u] + 1 then
                if DFS(Pair_V[v]) = true then
                    Pair_V[v] := u
                    Pair_U[u] := v
                    return true
        Dist[u] := inf
        return false
    return true

function Hopcroft-Karp is
    for each u in U do
        Pair_U[u] := NIL
    for each v in V do
        Pair_V[v] := NIL
    matching := 0
    while BFS() = true do
        for each u in U do
            if Pair_U[u] = NIL then
                if DFS(u) = true then
                    matching := matching + 1
    return matching
```

4.2 złożoność czasowa

Złożoność czasowa algorytmu Hopcroft–Karpa to $O(ElogV)$

4.3 Wykresy



Rysunek 3: liczba skojarzeń w zależności od k