Przedmiot: Projektowanie sieci telekomunikacyjnych
Prowadzący: Piotr Chołda piotr.cholda@agh.edu.pl
Kierunek: Elektronika i Telekomunikacja
Specjalność: Sieci i usługi
Semestr: II sem. (zimowy) studiów magisterskich

Konspekt

Piotr Chołda

12 października 2017

1 Algorytmy definiowane na grafach

1.1 Przykładowe problemy optymalizacji kombinatorycznej na grafach

1. Pojęcia programowania sieciowego i optymalizacji kombinatorycznej.
2. Problem poszukiwania minimalnego drzewa rozpinającego (MST, *minimum spanning tree*):
   - algorytm Kruskala:

```
1: procedure KRUSKAL(G = (V, E, f))
2:                                   ▷ Wyjściowe krawędzie MST: T
3:     T ← ∅
4:     E = E
5:     while |T| < |V| − 1 do
6:         e ← arg min_{i∈E} {f(i)}
7:             ▷ e: „najlżejsza" krawędź w zbiorze nieuwzględnionych krawędzi
8:         E ← E ∖ {e}
9:         if T ∪ {e} nie zawiera cyklu then
10:            T ← T ∪ {e}
11:        end if
12:    end while
13: end procedure
```

   - algorytm Prima (zwany również algorytmem Prima-Dijkstry):

```
1: procedure PRIM(G = (V, E, f), r ∈ V)
2:                                   ▷ Wyjściowe krawędzie MST: T
3:                                   ▷ r: korzeń
4:     T ← {r}
5:     V = V
6:     while V ≠ ∅ do
7:         L = {i ∈ E : i = {t, v}, t ∈ T, v ∈ V}
8:         e ← arg min_{i∈L} {f(i)}
9:             ▷ e: „najlżejsza" krawędź w zbiorze krawędzi, które powiększą drzewo
10:        V ← V ∖ {v ∈ V : e = {t, v}, t ∈ T}
11:        T ← T ∪ {e}
```

12: **end while**
13: **end procedure**

3. Pojęcie algorytmu zachłannego (*greedy algorithm*).

4. Przykłady „trudnych" problemów programowania sieciowego:
   - problem Steinera;
   - problem kolorowania (wierzchołków) w grafie; twierdzenie o czterech barwach, pojęcie grafu planarnego.

5. Algorytm przeszukiwania grafu wszerz (BFS, *Breadth-First Search*), drzewo ścieżek najkrótszych pod względem liczby przeskoków (*hops*):

1: **procedure** BFS($G = (V, A)$, $r \in V$)
2: ▷ r: korzeń
3: ▷ Inicjalizacja:
4: $\mathcal{S} \leftarrow \{r\}$
5: ▷ $\mathcal{S}$: zbiór wierzchołków, do których istnieje ścieżka skierowana od korzenia
6: $\mathcal{L} \leftarrow (r)$
7: ▷ $\mathcal{L}$: uporządkowana lista przeszukiwanych wierzchołków
8: $\mathcal{L}' \leftarrow V \setminus \{r\}$
9: ▷ $\mathcal{L}'$: zbiór dotychczas nieprzeszukanych wierzchołków
10: $predecessor(r) = 0$
11: ▷ $predecessor(j) = k$ oznacza, że poprzednikiem wierzchołka j na ścieżce skierowanej od korzenia r jest wierzchołek k
12: ▷ Korzeń nie ma poprzednika
13: ▷ Pętla główna:
14: **while** $\mathcal{L} \neq \emptyset$ **do**
15: **for all** $k \in \mathcal{L}$ **do**
16: **for all** $j \in \mathcal{L}'$ **do**
17: **if** $(k, j) \in A$ **then**
18: $\mathcal{S} \leftarrow \mathcal{S} \cup \{j\}$
19: $predecessor(j) = k$
20: $\mathcal{L} \leftarrow (\mathcal{L}, j)$
21: $\mathcal{L}' \leftarrow \mathcal{L}' \setminus \{j\}$
22: **end if**
23: **end for**
24: $\mathcal{L} \leftarrow \mathcal{L} \setminus \{k\}$
25: **end for**
26: **end while**
27: **return** ($\mathcal{S}$,$\mathcal{P}(\mathcal{S})$)
28: ▷ $\mathcal{P}(\mathcal{S})$: lista poprzedników wierzchołków należących do zbioru $\mathcal{S}$
29: **end procedure**

6. Algorytm przeszukiwania grafu w głąb (DFS, *Depth-First Search*):

1: **procedure** DFS($G = (V, A)$, $r \in V$)
2: ▷ r: korzeń
3: ▷ Inicjalizacja:
4: $\mathcal{S} \leftarrow \{r\}$

```
5:    ▷ S: zbiór wierzchołków, do których istnieje ścieżka skierowana od
      korzenia
6:    L' ← V ∖ {r}
7:                ▷ L': zbiór dotychczas nieprzeszukanych wierzchołków
8:    predecessor(r) = 0
9:                                                  ▷ Pętla główna:
10:   SearchDeep(r,G,S,L')
11:   return (S,P(S))
12: end procedure
```

Procedura wykonywana rekurencyjnie:

```
1: procedure SearchDeep(v,G,S,L')
2:     for all j ∈ L' do
3:         if (v, j) ∈ A then
4:             S ← S ∪ {j}
5:             predecessor(j) = v
6:             L' ← L' ∖ {j}
7:             SearchDeep(j,G,S,L')
8:         end if
9:     end for
10: end procedure
```

7. Problem poszukiwania przepływu maksymalnego. Twierdzenie Forda-Fulkersona. Algorytm rozwiązania problemu maksymalnego przepływu oparty na użyciu grafu rezydualnego (*residual graph*) i poszukiwaniu ścieżek powiększających (*augmenting flows*).

1.2 Zadania

- Proszę podać przykład takiego grafu ważonego G, który spełnia na raz wszystkie poniższe warunki:
  - ⋆ graf G jest spójny,
  - ⋆ wszystkie wagi są liczbami naturalnymi,
  - ⋆ graf G ma dziewięć wierzchołków,
  - ⋆ suma wag minimalnego drzewa rozpinającego grafu G jest równa połowie sumy wszystkich wag w tym grafie.
- Proszę pokazać, że minimalne drzewo rozpinające grafu pełnego K_9 o dowolnych wagach jest grafem dwudzielnym.

1.3 Lektury

1.3.1 Materiał wykładu

Zagadnienia omówione w ramach tego wykładu są w dużym stopniu opisane w następujących książkach:

- Maciej M. Sysło, Narsingh Deo, and Janusz S. Kowalik. *Algorytmy optymalizacji dyskretnej.* Wydawnictwo Naukowe PWN, Warszawa, 1999: rozdziały 3.4-3.6.

- Robin J. Wilson. *Wprowadzenie do teorii grafów*. Wydawnictwo Naukowe PWN, Warszawa, 2000: § 17, § 29.

1.3.2 Bibliografia uzupełniająca

- Ramesh Bhandari. *Survivable Networks. Algorithms for Diverse Routing*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999: przegląd różnych algorytmów przydatnych w projektowaniu sieci (niezawodnych).
- Andrew V. Goldberg and Robert E. Tarjan. Efficient Maximum Flow Algorithms. *Communications of the ACM*, 57(8):82–89, August 2014: problemy maksymalnego przepływu.
- Maciej M. Sysło, Narsingh Deo, and Janusz S. Kowalik. *Algorytmy optymalizacji dyskretnej*. Wydawnictwo Naukowe PWN, Warszawa, 1999: podstawy teoretyczne do naszego kursu.
- Robin J. Wilson. *Wprowadzenie do teorii grafów*. Wydawnictwo Naukowe PWN, Warszawa, 2000: zwięzłe wprowadzenie do teorii grafów, trochę algorytmów.