Problem
Min-cost Max-flow

Programowanie funkcyjne

Grzegorz Swatowski

09.01.2012

Problem Max-flow - definicja

Dana jest <u>**sieć przepływowa**</u> (G, c, s, t):

- **c : E -> R**$_{>=0}$ gdzie c((u, v)) nazywamy przepustowością krawędzi (u, v). Rozszerzamy notację: c : V*V -> R$_{>=0}$
- **s, t** - dwa wyróżnione wierzchołki, odp. źródło, ujście

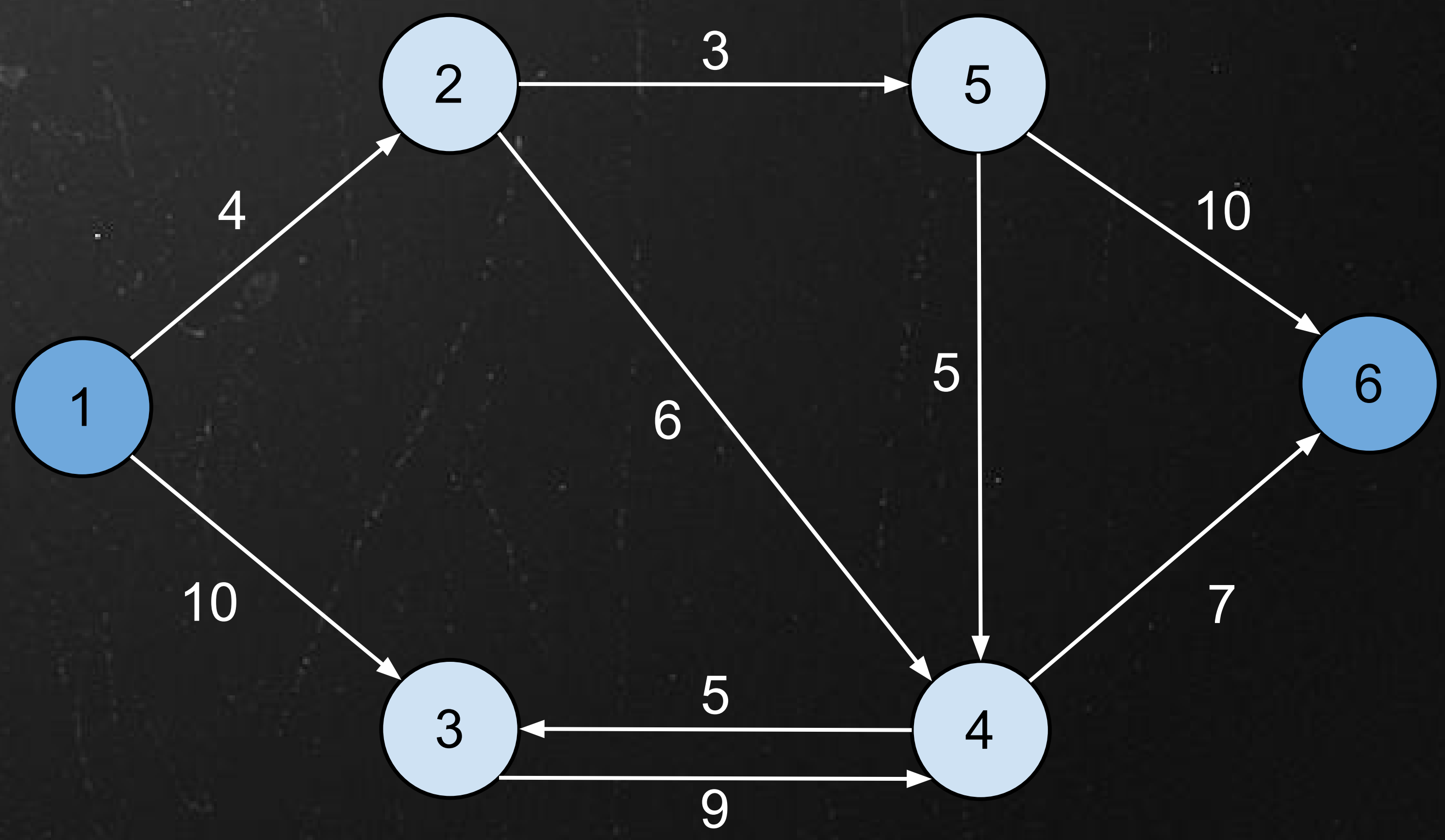
<u>**Przepływ**</u> sieci (G, c, s, t) to dowolna funkcja f : V*V->R t. że

- (**warunek przepustowości**)
  f(u, v) <= c(u, v) dla każdego u,v in V
- (**warunek skośnej symetryczności**)
  f(u, v) = - f(v, u) dla każdego u,v in V
- (**warunek zachowania przepływu**)
  sum(x in V, f(v, x)) = 0 dla każdego v in V/{s, t}

<u>**Wartość przepływu**</u>:
|f| = sum(x in V, f(s, x))

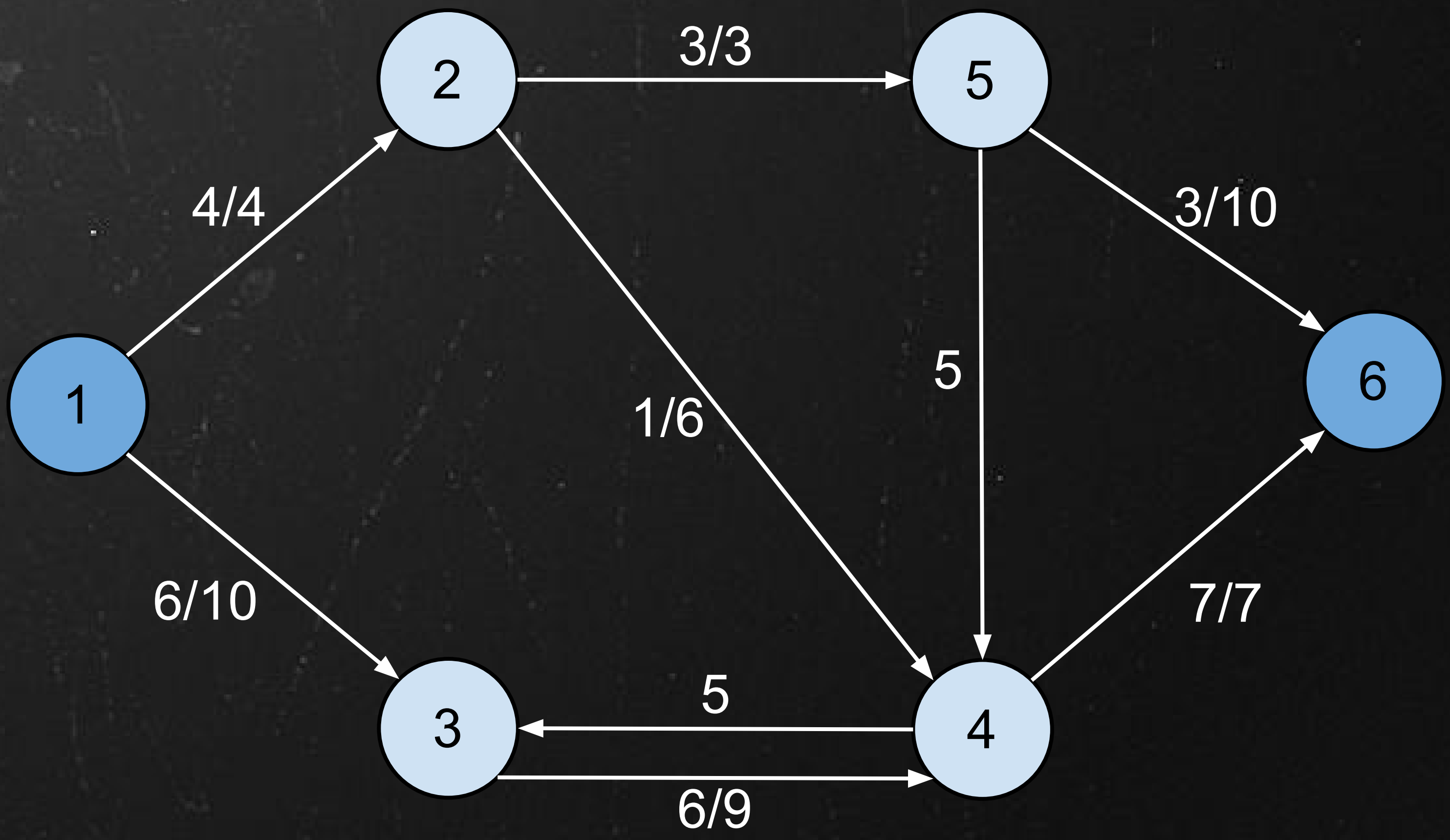
Problem Max-flow - przykład

Sieć przepływowa:



Problem Max-flow - przykład

Sieć przepływowa i przepływ:



$|f| = 10$

Problem Max-flow - zastosowania

- znajdowanie maksymalnego skojarzenia w grafie dwudzielnym
- znajdowanie najmniejszego pokrycia wierzchołkowego w grafie dwudzielnym (tw. Koniga-Egervary'ego)
- znajdowanie minimalnego przekroju
- znajdowanie największej liczby niezależnych wierzchołkowo ścieżek od s do t
- znajdowanie największej liczby niezależnych krawędziowo ścieżek od s do t

Problem Max-flow - złożoności algorytmów

- Algorytm Forda-Fulkersona O(|E| * |f*|)
- Algorytm Edmonsa-Karpa O(|V| * |E|^2)
- Algorytm Dinica O(|V|^2 * |E|)
- Algorytm trzech Hindusów O(|V|^3)

Problem Min-cost Max-flow - definicja

Dana jest **<u>sieć przepływowa</u>** (G, c, s, t, a)

- **c**, **s**, **t** jak wcześniej
- **a** : E -> $R_{>=0}$ gdzie a((x, y)) - koszt przesłania jednostki przepływu po (x, y)

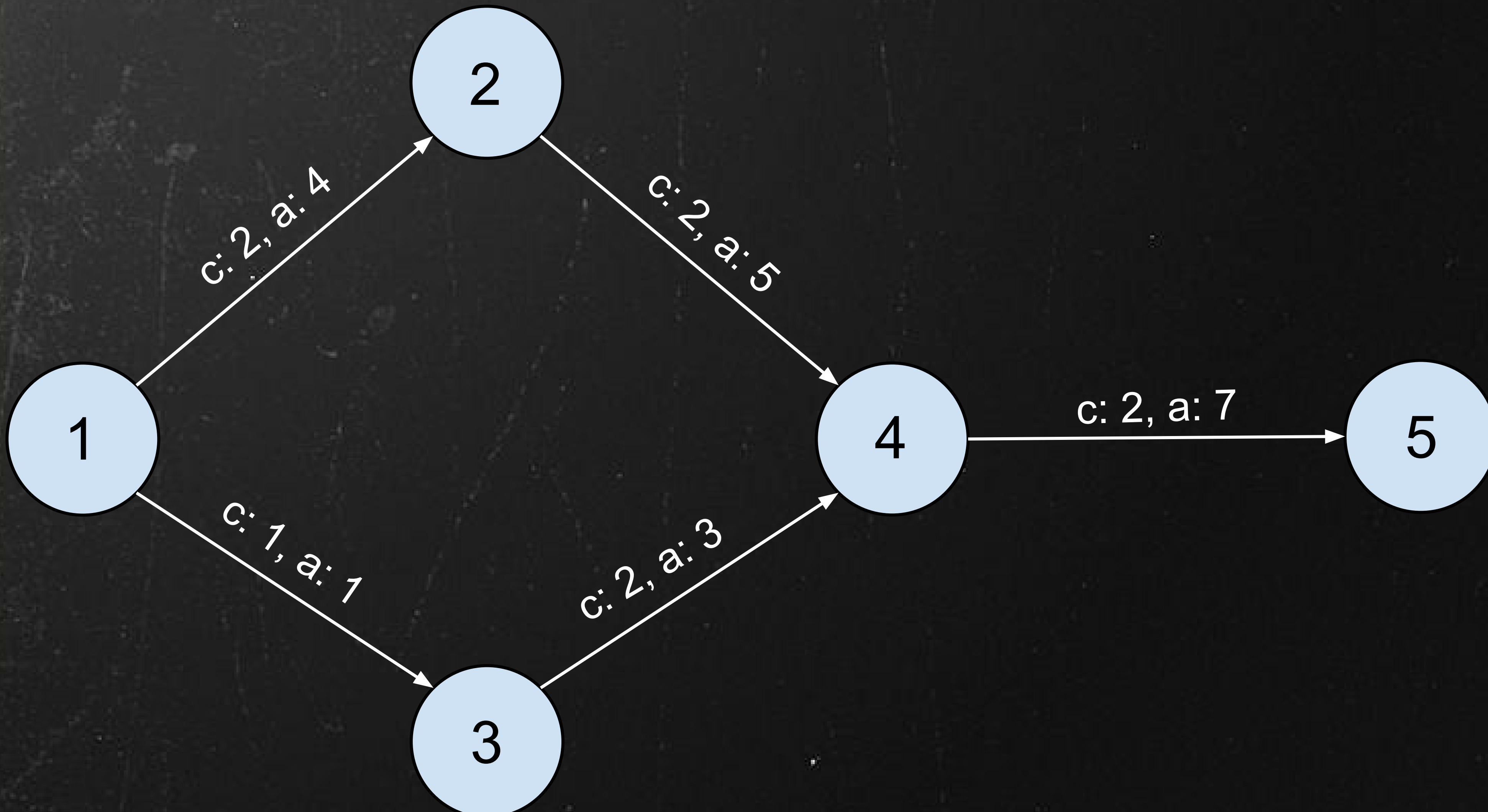
<u>Przepływ</u> sieci (G, c, s, t, a) to dowolna funkcja f : V*V->$R_{>=0}$ t. że :

- **(warunek przepustowości)**
  f(u, v) <= c(u, v) dla każdego u,v in V
- **(warunek zachowania przepływu)**
  sum((v, w) in E, f(v, w)) = sum((w, v) in E, f(w, v)) dla każdego w in V

<u>Koszt przepływu</u>:
cost(f) = sum(e in E, a(e) * f(e))

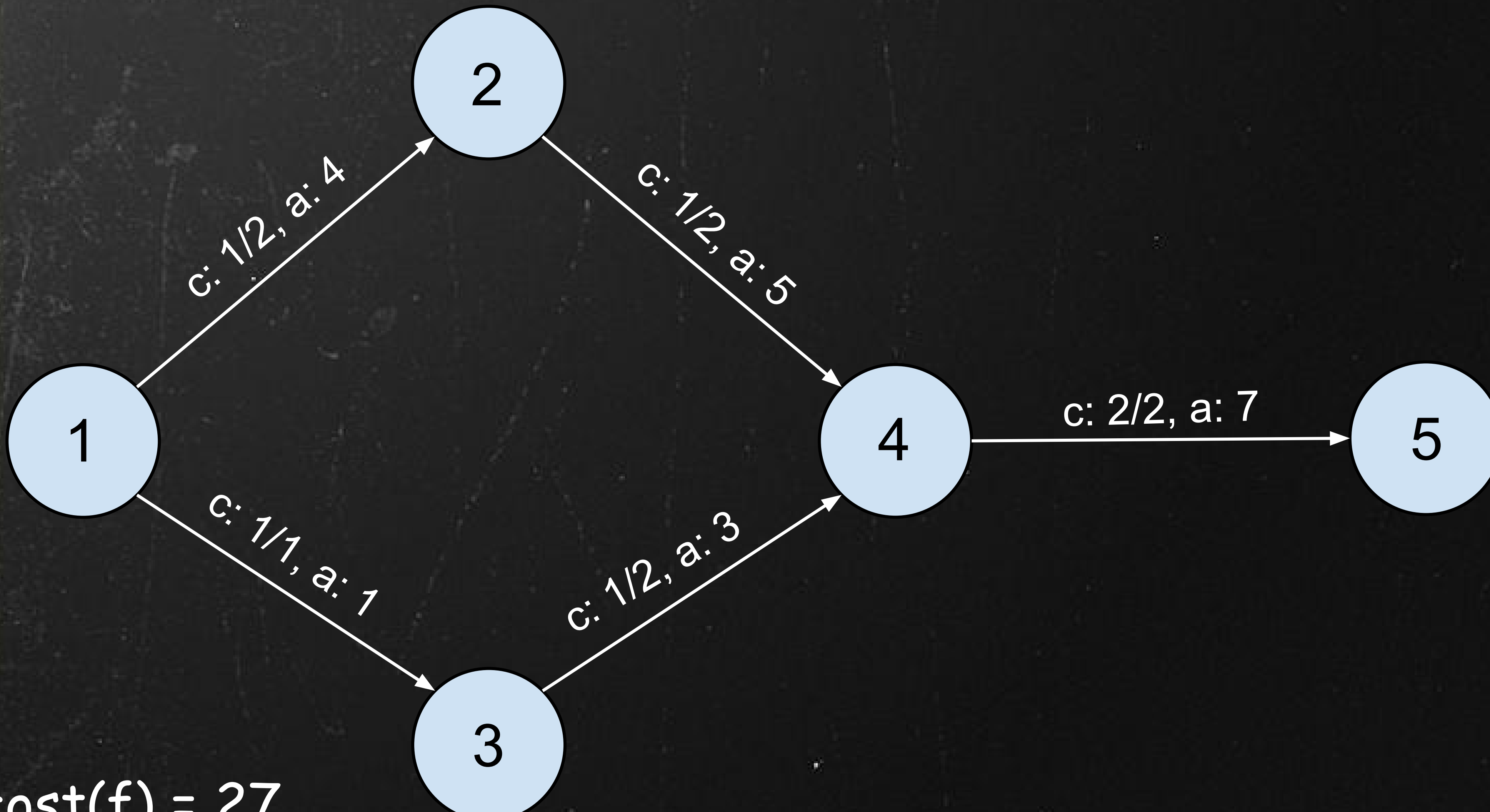
Problem Min-cost Max-flow - przykład

Sieć przepływowa:



Problem Min-cost Max-flow - przykład

Sieć przepływowa i przepływ:

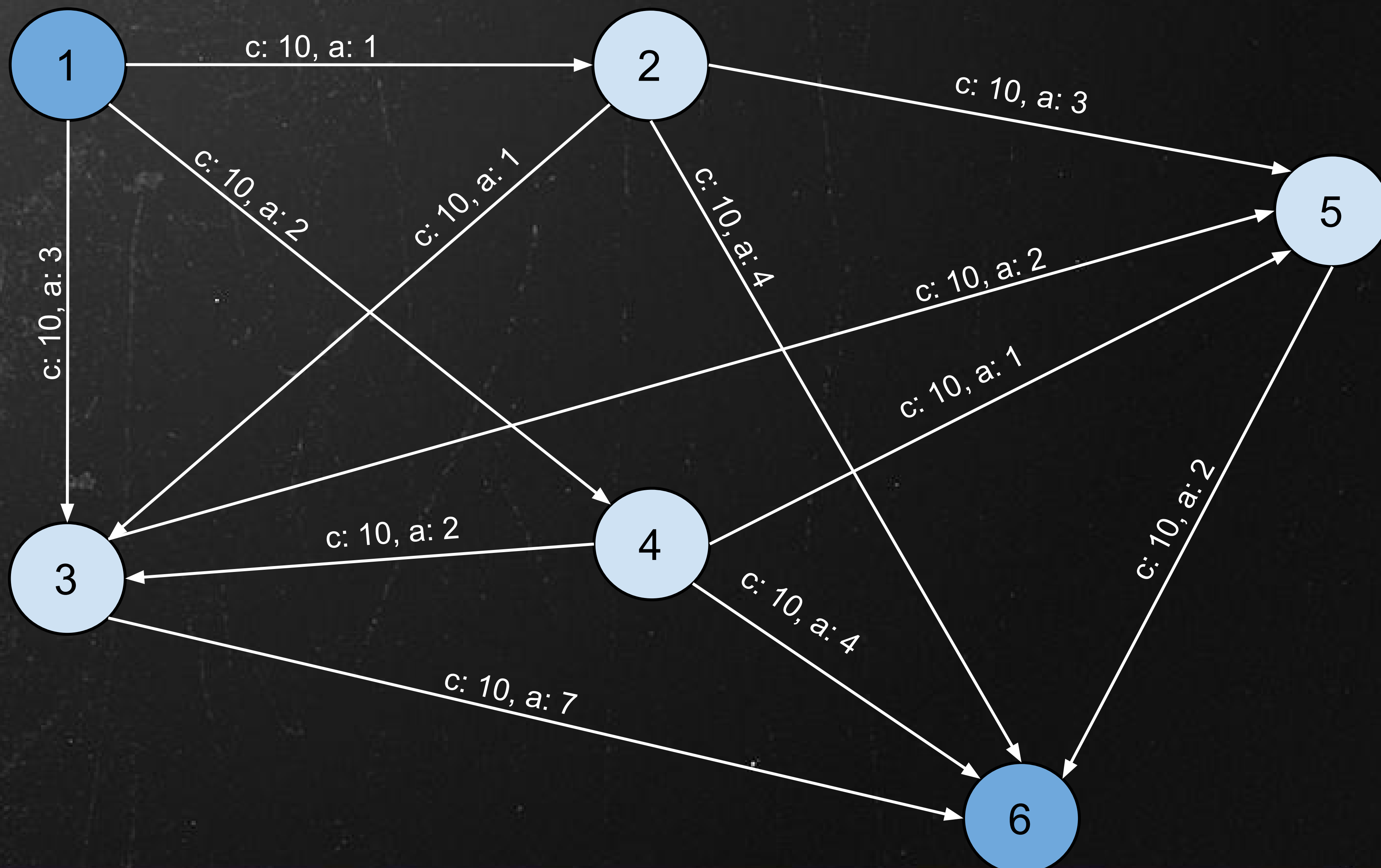


cost(f) = 27

|f| = 2

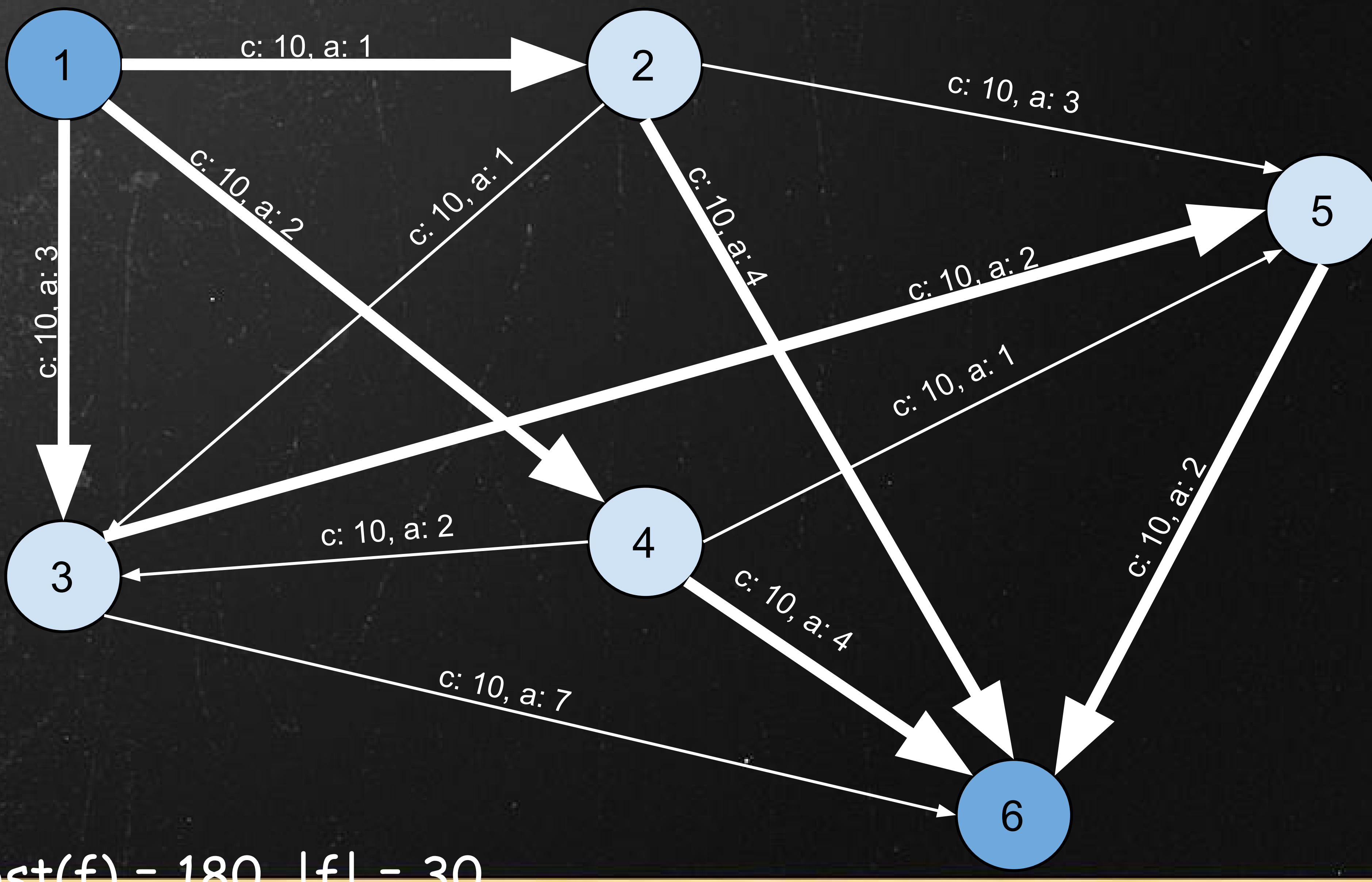
Problem Min-cost Max-flow - przykład

Sieć przepływowa:



Problem Min-cost Max-flow - przykład

Sieć przepływowa i przepływ:



cost(f) = 180, |f| = 30

Problem Min-cost Max-flow - algorytm

```
f <- 0 {pusty przepływ}
oblicz potencjał PI(v) := d(s,v) {obliczony algorytmem Bellmana-
                                  Forda}
dopóki istnieje ścieżka powiększająca:
    { PI jest potencjałem }
    P <- najtańsza ścieżka powiększająca {Alg. Dijkstry
         w grafie z wagami a_pi(u,v) = a(u,v) + PI(u) - PI(v). To będzie
         najtańsza ścieżka także względem a}
    powiększ f wzdłuż P
    for x in V:
        PI(x) := PI(x) + d_pi(s,x)
        { d_pi - odległość w sensie wag a_pi obliczona alg. Dijkstry}
```

Problem Min-cost Max-flow - złożoność

```
f <- 0 {pusty przepływ}
oblicz potencjał PI(v) := d(s,v) {obliczony algorytmem Bellmana-
                                  Forda}            O(|E|*|V|)
dopóki istnieje ścieżka powiększająca:              O(|f*|)
    { PI jest potencjałem }
    P <- najtańsza ścieżka powiększająca {Alg. Dijkstry
         w grafie z wagami a_pi(u,v) = a(u,v) + PI(u) - PI(v). To będzie
         najtańsza ścieżka także względem a}    O(|V|*log|V|+|E|)
    powiększ f wzdłuż P                          O(|E|)
    for x in V:                                  O(|V|)
        PI(x) := PI(x) + d_pi(s,x)
        { d_pi - odległość w sensie wag a_pi obliczona alg. Dijkstry}
```

Całość: O(|V|*|E| + |f*|*(|V|*log|V| + |E|))

Problem Min-cost Max-flow - moduły

```
f <- 0 {pusty przepływ}
oblicz potencjał PI(v) := d(s,v) {obliczony algorytmem Bellmana-
                                   Forda}
dopóki istnieje ścieżka powiększająca:
    { PI jest potencjałem }
    P <- najtańsza ścieżka powiększająca {Alg. Dijkstry
         w grafie z wagami a_pi(u,v) = a(u,v) + PI(u) - PI(v). To będzie
         najtańsza ścieżka także względem a}
    powiększ f wzdłuż P
    for x in V:
        PI(x) := PI(x) + d_pi(s,x)
        { d_pi - odległość w sensie wag a_pi obliczona alg. Dijkstry}
```

Problem Min-cost Max-flow - moduły

- Moduł szukający najkrótszą ścieżkę
  - Algorytm Dijkstry
    - kolejka priorytetowa (zaimplementowana na AVLu)
  - Algorytm Bellmana Forda
- Mapa (zaimplementowana na AVLu)
- Kolejka priorytetowa (zaimplementowana na AVLu)
- Moduł rozwiązujący problem Min-cost Max-Flow

Problem Min-cost Max-flow - funktory

- SHORTEST_PATH_SEEKER
  = functor(Queue : PRIORITY_QUEUE) -> functor(AvlMap : MAP)
- PRIORITY_QUEUE = functor(Order : LINEAR_ORDER)
- MAP = functor(Order : LINEAR_ORDER_TUPLE)
- MAX_FLOW_MIN_COST
  = functor(AvlMap : MAP) ->
  functor(Dijkstra : SHORTEST_PATH_FINDER) ->
  functor(BellmanFord : SHORTEST_PATH_FINDER)

Dziękuję
Pytania?