Wyznaczanie maksymalnego przepływu

Bibliteka Graph zawiera klasę statyczną Flows udostępniającą metody pozwalające na znalezienie maksymalnego przepływu.

Metoda FordFulkerson wyznacza maksymalny przepływ od wierzchołka source do wierzchołka target posługując się algorytmem Forda-Fulkersona (z wyborem ścieżki powiększającej przez przeszukiwanie w głąb). Metoda zwraca parę (wartość maksymalnego przepływu, maksymalny przepływ), gdzie maksymalny przepływ zadany jest jako graf skierowany ważony.

Przykład wywołania metody:

```
var (flowValue, f) = Flows.FordFulkerson(G, s, t);
```

Maksymalny przepływ o minimalnym koszcie

Biblioteka udostępnia klasę NetworkWithCosts<TCapacity, TCost>, która reprezentuje sieć przeływową z kosztami, gdzie przepustowości krawędzi są typu TCapacity, a koszty typu TCost. Klasa ta dziedziczy po DiGraph<(TCapacity, TCost)>.

Maksymalny przepływ o minimalnym koszcie można wyznaczyć, posługując się metodą MinCostMaxFlow zdefiniowaną w klasie Flows.

Przykład wywołania metody:

```
NetworkWithCosts<int, double> G = new NetworkWithCosts<int, double>(10);
G.AddEdge(0, 1, 5, 3.5);
var (flowValue, flowCost, f) = Flows.MinCostMaxFlow(G, s, t);
```

Zadanie: Spójność krawędziowa i przekrój

Spójność krawędziowa grafu to minimalna liczba krawędzi, którą trzeba usunąć by rozspójnić graf. Ważona spójność krawędziowa to minimalna suma wag krawędzi które należy usunąć by rozspójnić graf.

Niech s, t będą wierzchołkami grafu. $s-t$-przekrój tego grafu to zbiór krawędzi taki, że po jego usunięciu wierzchołki s i t znajdują się w dwóch różnych spójnych składowych grafu. Wartością $s-t$ przekroju jest suma wag krawędzi wchodzących w jego skład.

Celem zadania jest napisanie dwóch metod rozszerzających interfejs Graph.

1. MinCut(int s, int t, out Edge<double>[] cut) - zwracającą minimalną wartość s-t-przekroju.

   Parametr wyjściowy cut powinien zawierać zbiór krawędzi realizujący minimalny przekrój (metoda przyjmuje tylko grafy ważone nieskierowane).

2. EdgeConnectivity(out Edge<double>[] cutingSet) - zwracającą ważoną spójność krawędziową.

   Parametr wyjściowy cutingSet powinien zawierać zbiór krawędzi rozspójniający graf o minimalnej wadze (metoda przyjmuje tylko grafy ważone nieskierowane).

Wymagania złożonościowe:

- punkt 1 powinien działać w czasie MF(n, m),
- punkt 2 powinien działać w czasie n*MF(n, m),

gdzie n to liczba wierzchołków, a MF(n, m) to złożoność algorytmu znajdującego maksymalny przepływ.

Wskazówki:

1. Do wszystkich metod należy wykorzystać algorytm maksymalnego przepływu. Wartość maksymalnego przepływu w grafie jest równa wartości minimalnego przekroju.
2. Grafy nieskierowane (na takich mają działać implementowane metody) należy przekształcić do odpowiadających im grafów skierowanych.
3. Po usunięciu z grafu G $s-t$ przekroju w grafie G powstają dwie spójne składowe, nazwijmy je S i T.

   S to spójna składowa zawierająca wierzchołek s, a T to spójna składowa zawierająca wierzchołek t. Zbiór S można wyznaczyć łatwo z sieci rezydualnej (dla maksymalnego przepływu) – jest to zbiór wierzchołków osiągalnych (w sieci rezydualnej) z wierzchołka s (a T to zbiór wierzchołków nieosiągalnych).

   $s-t$ przekrój to zbiór krawędzi, których jeden koniec jest w zbiorze S, a drugi w T.
4. Wygodnie jest napisać metodę pomocniczą wyznaczającą przekrój dla danego grafu i danego maksymalnego przepływu w tym grafie
5. Jak wyznaczyć ważoną spójność krawędziową? Odpowiedź: Popatrz na wymaganą złożoność.

Punktacja:

- MinCut – 1.5 pkt.
- EdgeConnectivity
  - Wartość ważonej spójności krawędziowej – 0.5 pkt.
  - Zbiór rozspójniający – 0.5 pkt.

Zadanie: planowanie produkcji

Pliki w zadaniu:

- Plik Program.cs zawiera testy opracowanych rozwiązań. Nie należy modyfikować zawartości tego pliku.
- Plik DataStructures.cs zawiera pomocnicze struktury danych dla zadania. Nie należy modyfikować zawartości tego pliku.
  - Struktura PlanData używana jest do przekazywania danych wejściowych dotyczących produkcji, magazynu i sprzedaży. W tym kontekście pole Quantity oznacza limity produkcji, magazynu lub sprzedaży, zaś pole Value – odpowiednio koszt produkcji, koszt przechowania w magazynie lub cenę sprzedaży jednej sztuki.

    Ponadto struktura wykorzystywana jest do zwrócenia danych wyjściowych z pisanych funkcji (liczby wyprodukowanych sztuk telewizorów i otrzymanego zysku). Liczbę telewizorów umieszczamy w polu Quantity, zaś zysk – w pole Value.
  - Struktura SimpleWeeklyPlan, używana w części 1., przechowuje liczbę sztuk wyprodukowanych (UnitsProduced), sprzedanych (UnitsSold) i przechowanych w magazynie (UnitsStored) w każdym tygodniu jako wartości liczbowe całkowite.
  - W strukturze WeeklyPlan, używanej w części 2., sprzedaż (UnitsSold) określana jest w postaci tablicy wartości całkowitych. Kolejne elementy tej tablicy oznaczają liczbę sztuk sprzedanych kontrahentowi o danym numerze w danym tygodniu. Liczba sztuk wyprodukowanych i przechowanych w magazynie jest reprezentowana identycznie, jak w strukturze SimpleWeeklyPlan.
- W pliku ProductionPlanner.cs powinno znaleźć się rozwiązanie zadania.

Część 1

Fabryka telewizorów opracowuje plan produkcji i sprzedaży. Produkcja w fabryce odbywa się zgodnie z następującymi założeniami:

- Fabryka w ciągu danego tygodnia może wyprodukować ograniczoną liczbę telewizorów (określoną w polu Quantity). Limit produkcji zmienia się z tygodnia na tydzień.
- Wyprodukowanie jednej sztuki telewizora wiąże się z ustalonym kosztem (określonym w polu Value). Koszt produkcji także zmienia się z tygodnia na tydzień.
- Fabryka sprzedaje telewizory jednemu kontrahentowi. Kontrahent ma ograniczone zapotrzebowanie na towar i kupuje sztuki telewizorów po określonej cenie. Zarówno zapotrzebowanie (Quantity), jak i cena sprzedaży telewizorów (Value), zmieniają się z tygodnia na tydzień.
- W razie potrzeby telewizory mogą być przechowane w magazynie, w celu późniejszej sprzedaży. Magazyn ma ograniczoną, stałą pojemność (Quantity). Przechowanie jednej sztuki w magazynie przez tydzień wiąże się z niezmiennym kosztem magazynowania (Value).

Twoim celem jest wyznaczenie takiego planu produkcji, który maksymalizuje produkcję fabryki (Quantity), a dla tej maksymalnej produkcji maksymalizuje zysk fabryki (Value – różnicę przychodów ze sprzedaży i kosztów produkcji oraz przechowywania towarów).

Oprócz podania osiągniętego zysku, należy podać szczegóły planu w poszczególnych tygodniach produkcji, w postaci tablicy obiektów WeeklyPlan (ile sztuk wyprodukowano - UnitsProduced, ile sprzedano - UnitsSold, ile przechowano w magazynie - UnitsStored).

Część 2

Po pomyślnym rozplanowaniu sprzedaży fabryka rozszerza swoją działalność i podpisuje umowy z wieloma kontrahentami.

- Założenia dotyczące limitów i kosztów produkcji pozostają takie same.
- Fabryka dalej może przechowywać telewizory w magazynie, na tych samych zasadach, co wcześniej.
- Fabryka może jednak wybierać, któremu kontrahentowi chce sprzedać ile sztuk telewizorów w danym tygodniu. Każdy kontrahent ma swoje zapotrzebowanie na telewizory, oraz swoją cenę zakupu. Tak jak w części 1., oba parametry zmieniają się z tygodnia na tydzień u każdego z kontrahentów.

Tym razem, Twoim celem jest wyznaczenie planu sprzedaży maksymalizującego wyłącznie zysk fabryki. Warto zauważyć, że czasami zmniejszenie produkcji może skutkować większym zyskiem.

W szczegółowym planie sprzedaży należy określić, ile sztuk sprzedano poszczególnym kontrahentom w danym tygodniu, odpowiednio wypełniając pole UnitsSold, które w tym przypadku jest tablicą. Ponadto, podobnie jak w części 1., należy określić liczbę sztuk wyprodukowanych i sprzedanych w każdym tygodniu.

Oczekiwana złożoność obliczeniowa rozwiązania zależy wyłącznie od liczby tygodni i kontrahentów. W szczególności oznacza to, że jest ona niezależna od maksymalnego zysku i wyznaczonej liczby wyprodukowanych telewizorów.

Wskazówka:

W porównaniu do sieci maksymalizującej w pierwszej kolejności produkcję a w drugiej kolejności zyski (jak w części pierwszej), sieć maksymalizująca tylko zyski (i nie koniecznie maksymalizująca produkcję) powinna mieć dodatkowe wierzchołki i dodatkowe krawędzie o kosztach zerowych. A przy konstrukcji optymalnego planu produkcji część przepływu należy zignorować"

Uwagi

W obu etapach przyjmujemy następujące założenia dotyczące danych wejściowych:

- Liczba tygodni w planie jest dodatnia.
- Limit produkcji i koszt produkcji jest nieujemny w każdym tygodniu.
- Limit sprzedaży i cena sprzedaży jest nieujemna w każdym tygodniu.
- Liczba sztuk przechowywanych w magazynie oraz koszt przechowania są nieujemne.

Dodatkowo w drugim etapie zakładamy, że liczba klientów jest dodatnia.

Funkcje do wyznaczania planu w części domowej powinny weryfikować poprawność argumentów i w razie naruszenia jednego z powyższych warunków zgłaszać wyjątek ArgumentException.

W klasie ProductionPlanner znajduje się właściwość ‘ShowDebug‘, która pozwala przy testach wyznaczających plan produkcji wypisać na konsolę szczegóły skonstruowanego planu.

Punktacja

- Część 1, wyznaczenie maksymalnej produkcji i zysku – 1 pkt.
- Część 1, wyznaczenie planu produkcji – 0,5 pkt.
- Część 2, wyznaczenie maksymalnej produkcji i zysku – 0,5 pkt.
- Część 2, wyznaczenie planu produkcji – 0,5 pkt.