



AKADEMIA GÓRNICZO-HUTNICZA  
IM. STANISŁAWA STASZICA W KRAKOWIE

Akademia Górniczo-Hutnicza w Krakowie  
Wydział Fizyki i Informatyki Stosowanej  
Informatyka Stosowana  
6 maja 2022

Systemy równoległe i rozproszone

## Minimalne drzewo rozpinające – algorytm Prim’a

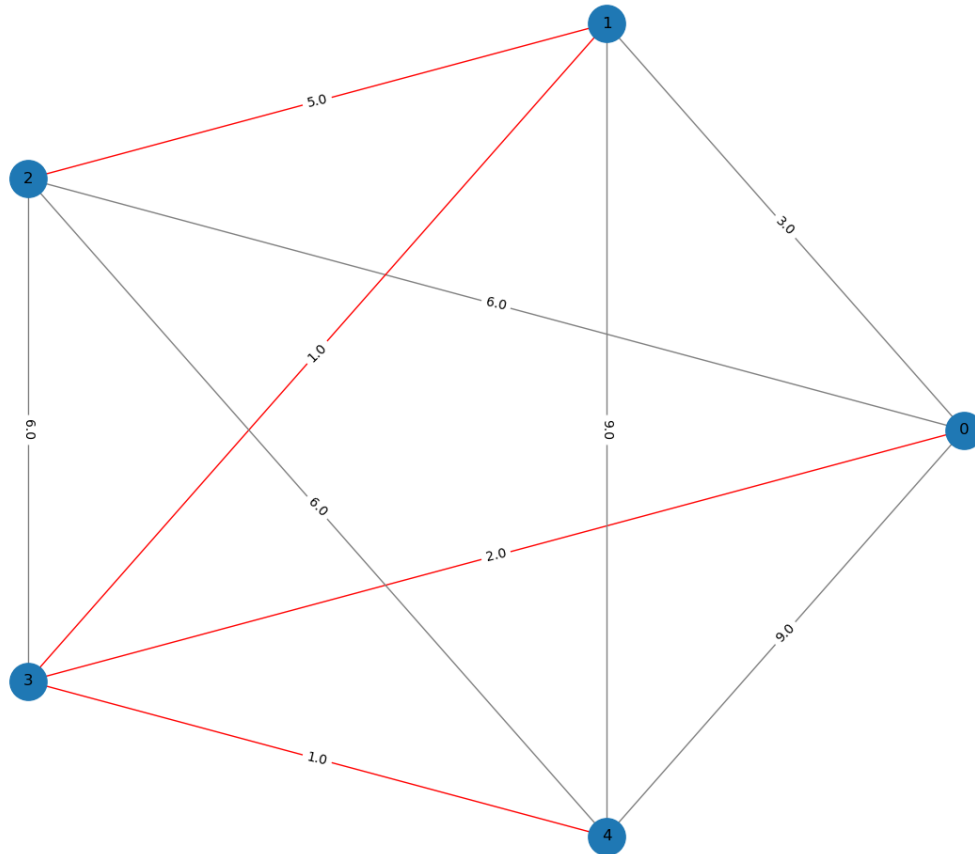
Skład zespołu:  
Bartosz Rogowski  
Aleksandra Rolka

### Spis treści

1	Opis problemu	2
2	Najważniejsze zmienne używane w programie	3
3	Schemat blokowy programu	4
4	Obsługa programu	6
	Bibliografia	6

# 1 Opis problemu

Jednym z często przeprowadzanych na grafach operacjach jest znalezienie ich minimalnych drzew rozpinających. Minimalne drzewo rozpinające (ang. *minimal spanning tree* – MST) grafu  $G$  to drzewo (acykliczny spójny graf nieskierowany) rozpinające (czyli zawierające wszystkie wierzchołki grafu  $G$ ), mające  $N - 1$  krawędzie o najmniejszych wagach.

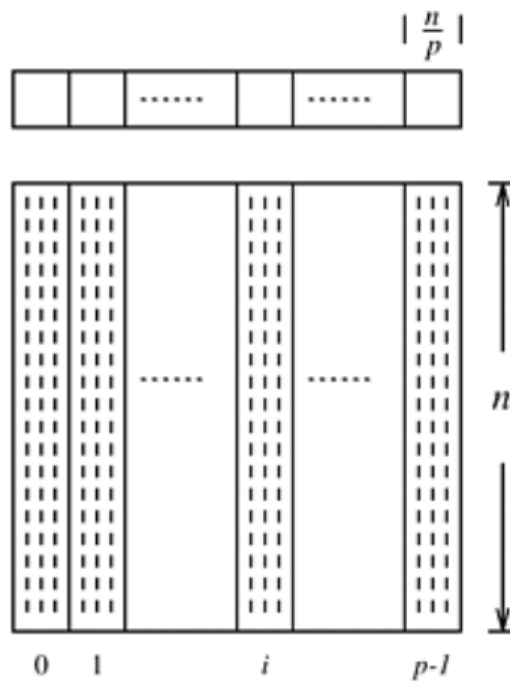


Rysunek 1: Przykład minimalnego drzewa rozpinającego (krawędzie zaznaczone na czerwono) na grafie

Algorytm Prima pozwala znaleźć MST dla danego za pomocą macierzy sąsiedztwa grafu. Wykorzystuje podejście zachłanne: zaczynając od ustalonego wierzchołka (najprościej od tego z indeksem 0), z listy sąsiadów dostępnych (wierzchołków do których można dojść) z odwiedzonych już wierzchołków wybierana jest krawędź o najmniejszej wadze prowadząca do innego nieodwiedzonego jeszcze wierzchołka. Algorytm przechodzi dalej, aż do wybrania  $N - 1$  krawędzi tworzących MST.

Zrównoleglenie tego programu jest możliwe dzięki podzieleniu macierzy<sup>1</sup> sąsiedztwa grafu na mniejsze podmacierze dla każdego procesu. Funkcja `MPI_Scatterv` umożliwia owy podział [2]. Podmacierze te są zapisywane jako tablice jednowymiarowe. Na rys. 2 przedstawiono takie rozdzielenie, np. proces 0 dostał trzy pierwsze kolumny, proces 1 – kolejne trzy itd. Każdy proces sprawdza krawędzie prowadzące do nieodwiedzonych jeszcze sąsiadów z wierzchołka  $k$  ( $k = 0, 1, \dots, N - 1$ ) i lokalnie wybiera tę o najmniejszej wadze. Następnie za pomocą `MPI_Allreduce` wyłaniana jest najmniejsza waga i wierzchołek, do którego prowadzi owa krawędź. Informacje o wierzchołkach pomiędzy znaną optymalną krawędzią są propagowane do

<sup>1</sup>W naszym przypadku przed dostarczeniem do odpowiedniej metody, macierz ta jest „spłaszczana” do postaci jednowymiarowej.



Rysunek 2: Podział macierzy na podmacierze zapisywane w postaci wektorów (źródło: [1])

wszystkich procesów poprzez `MPI_Bcast`, a do zmiennej przechowującej sumę wag dodawana jest nowo znaleziona.

## 2 Najważniejsze zmienne używane w programie

- `rank` – numer procesu
- `procNum` – całkowita liczba procesów używana w programie
- `graph` – graf zapisany w postaci macierzy sąsiedztwa
- `chunkSize` – liczba kolumn macierzy `graph`, które zostaną przesłane; `chunkSize` jest najpierw obliczany dla każdego procesu jako  $\lfloor \frac{N}{\text{procNum}} \rfloor$ , a następnie od procesu 0 dodawane są kolumny wynikające z użycia liczby procesów niepodzielnej przez wymiar macierzy
- `displs` – przesunięcie – mówi, w której kolumnie macierzy `graph` zaczyna się podmacierz `chunk`
- `chunk` – tablica, do której zapisywane są fragmenty macierzy `graph`, do każdego procesu przekazywana jest inna część
- `MST` – `graph` z wagami tylko dla tych wierzchołków, które tworzą minimalne drzewo rozpinające
- `mstEdges` – tablica, której indeksy to wierzchołki początkowe, a wartości to wierzchołki końcowe krawędzi

Przykład:  
załóżmy, że:

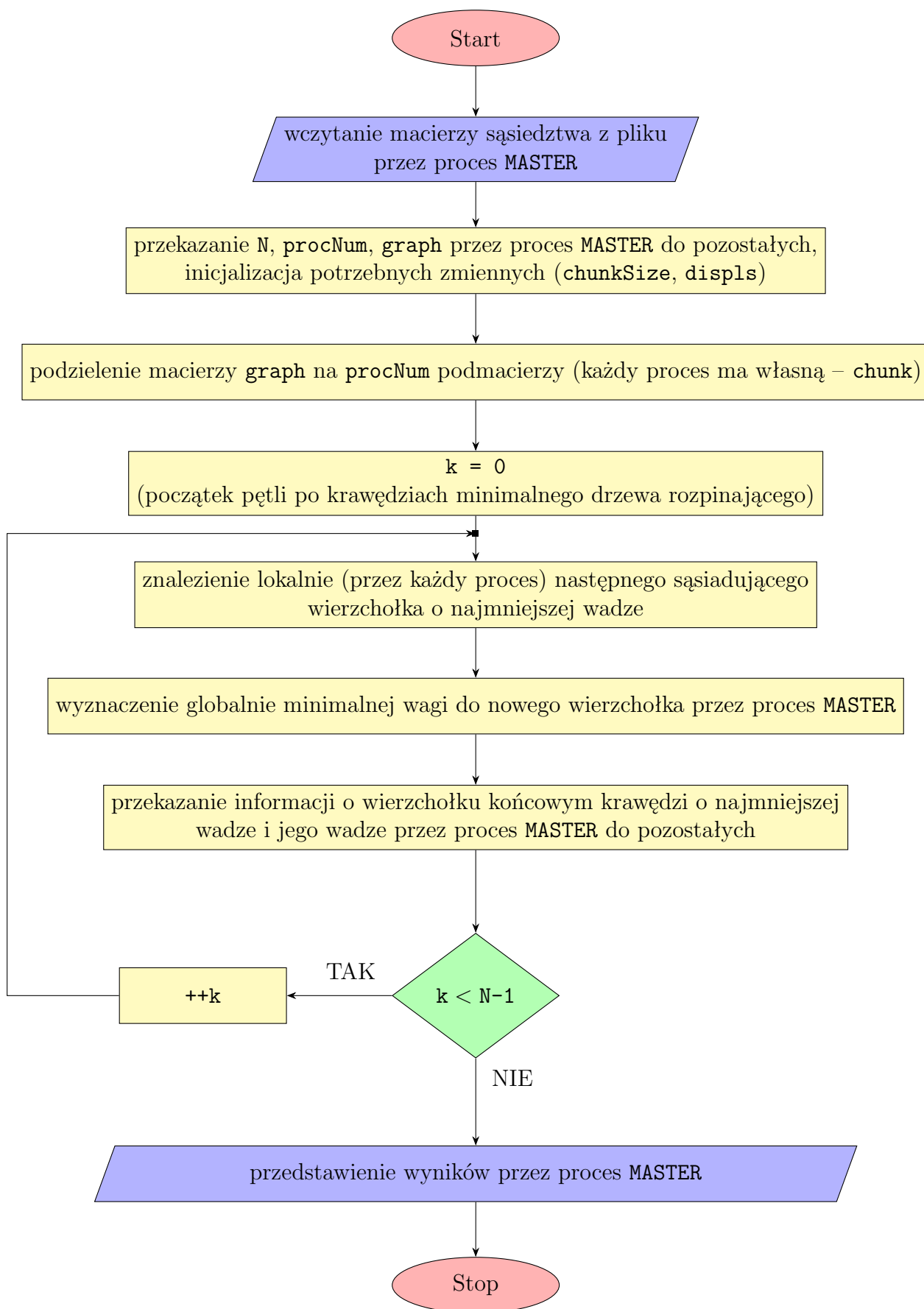
- `procNum = 3`
- $\text{graph} = \begin{bmatrix} 0 & 3 & 6 & 2 & 9 \\ 3 & 0 & 5 & 1 & 9 \\ 6 & 5 & 0 & 6 & 6 \\ 2 & 1 & 6 & 0 & 1 \\ 9 & 9 & 6 & 1 & 0 \end{bmatrix},$

wówczas:

- `chunkSize = { 2, 2, 1 }`
- `displs = { 0, 2, 4 }`
- dla procesu o numerze `rank` równym 0: `chunk = { 0, 3, 6, 2, 9, 3, 0, 5, 1, 9 }`
- dla procesu o numerze `rank` równym 1: `chunk = { 6, 5, 0, 6, 6, 2, 1, 6, 0, 1 }`
- dla procesu o numerze `rank` równym 2: `chunk = { 9, 9, 6, 1, 0 }`

### 3 Schemat blokowy programu

Schemat blokowy programu został przedstawiony na rys. 3.



Rysunek 3: Schemat blokowy programu

## 4 Obsługa programu

Program przyjmuje jeden obowiązkowy argument, którym jest ścieżka do pliku z podaną macierzą sąsiedztwa (wartości oddzielone spacjami, każdy wiersz w nowej linii) oraz jeden opcjonalny argument, do którego można zapisać macierz sąsiedztwa reprezentującą wynikowe minimalne drzewo rozpinające. Program wypisuje sumę wag MST oraz czas wykonywania algorytmu Prima dla każdego z procesów.

Do uruchomienia programu trzeba przygotować środowisko MPI, a następnie skompilować program. W tym celu wykorzystano plik `makefile`, który wykonuje niezbędne w tym celu operacje.

- `make prepare` – przygotowuje środowisko
- `make` – kompiluje program
- `make run` – uruchamia program, potrzebuje podania argumentów:
  - `n` – liczba procesów
  - `in` – ścieżka do pliku z macierzą sąsiedztwa grafu<sup>1</sup> (w folderze `example` znajdują się przygotowane pliki wejściowe)
  - `out` (opcjonalnie) – ścieżka do pliku z macierzą wyjściową MST
- `make run0_` – przygotowane komendy z dostowanymi argumentami (`_` oznacza cyfrę od 1 do 5)
- `make clean` – przywraca folder do stanu wejściowego (usuwa pliki wynikowe, wykonywalny etc.)
- `make mem` – walidacja programu za pomocą narzędzia Valgrind

## Bibliografia

- [1] Ananth Grama et al. *Introduction to parallel computing*. Pearson Education, 2003, pp. 442–444.
- [2] *MPI Scatterv*. URL: [https://www.mpich.org/static/docs/v3.1/www3/MPI\\_Scatterv.html](https://www.mpich.org/static/docs/v3.1/www3/MPI_Scatterv.html). (data dostępu: 2 maja 2022).

---

<sup>1</sup>Uwaga: Graf musi być spójny i nieskierowany. Zakłada się, że dostarczany plik jest prawidłowy.