

# *Programowanie równoległe i rozproszone:*

## Projekt nr 2 Algorytm poszukiwań w głąb (depth-first) The 8-puzzle problem

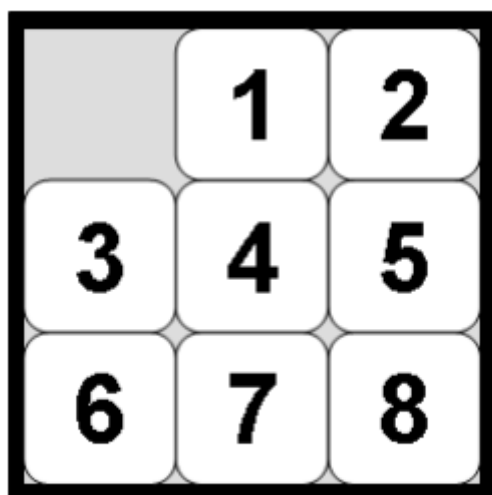
Autor: Gabriel Naleźnik, Adrian Góral  
20.06.2022 r.

### 1. Wstęp

Celem pierwszego projektu realizowanego w ramach przedmiotu: „programowanie równoległe i rozproszone”, było napisanie równoległego algorytmu poszukiwań w głąb (DFS) rozwiązującego zadanie „8-puzzle problem”.

### 2. Problem

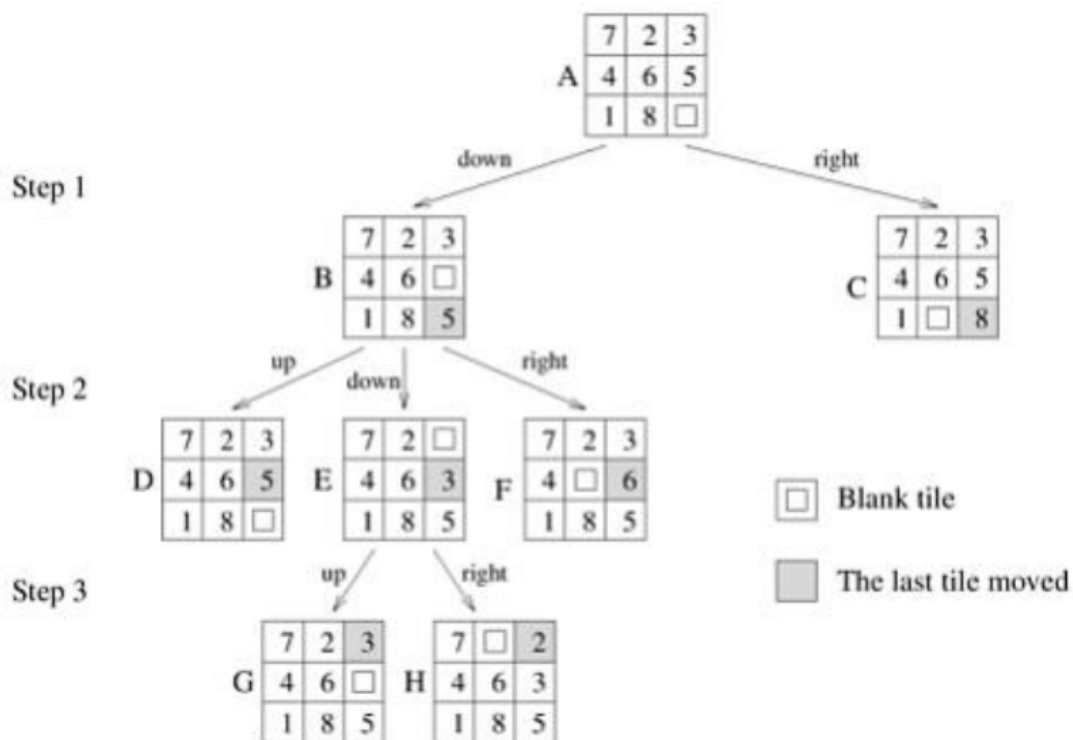
8-puzzle problem (pot. „Przesuwanka”) - układanka zbudowana z pola, w którym znajduje się 8 kwadratowych klocków o jednakowych rozmiarach ułożonych w kwadrat 3×3 i ponumerowanych od 1 do 8. Jedno miejsce jest puste i umożliwia przesuwanie sąsiednich klocków względem siebie. Celem gry jest ułożenie klocków w określony sposób, najczęściej w porządku rosnącym od 1 do 8 bądź innym określonym warunkami zadania, przez przesuwanie ich względem siebie. Możliwe są również inne układy, jak również zastąpienie liczb rysunkiem lub hasłem słownym. Zamiana klocków miejscami jest niedozwolona.



Rys 1. Końcowy poprawny układ

### 3. Algorytm

Algorytm przeszukiwania w głąb (ang. Depth-first search) - algorytm przeszukiwania grafu. Przeszukiwanie w głąb polega na badaniu wszystkich krawędzi wychodzących z podanego wierzchołka. Po zbadaniu wszystkich krawędzi wychodzących z danego wierzchołka algorytm wraca do wierzchołka, z którego dany wierzchołek został odwiedzony.



Rys 2. Drzewo przeszukiwań w głąb

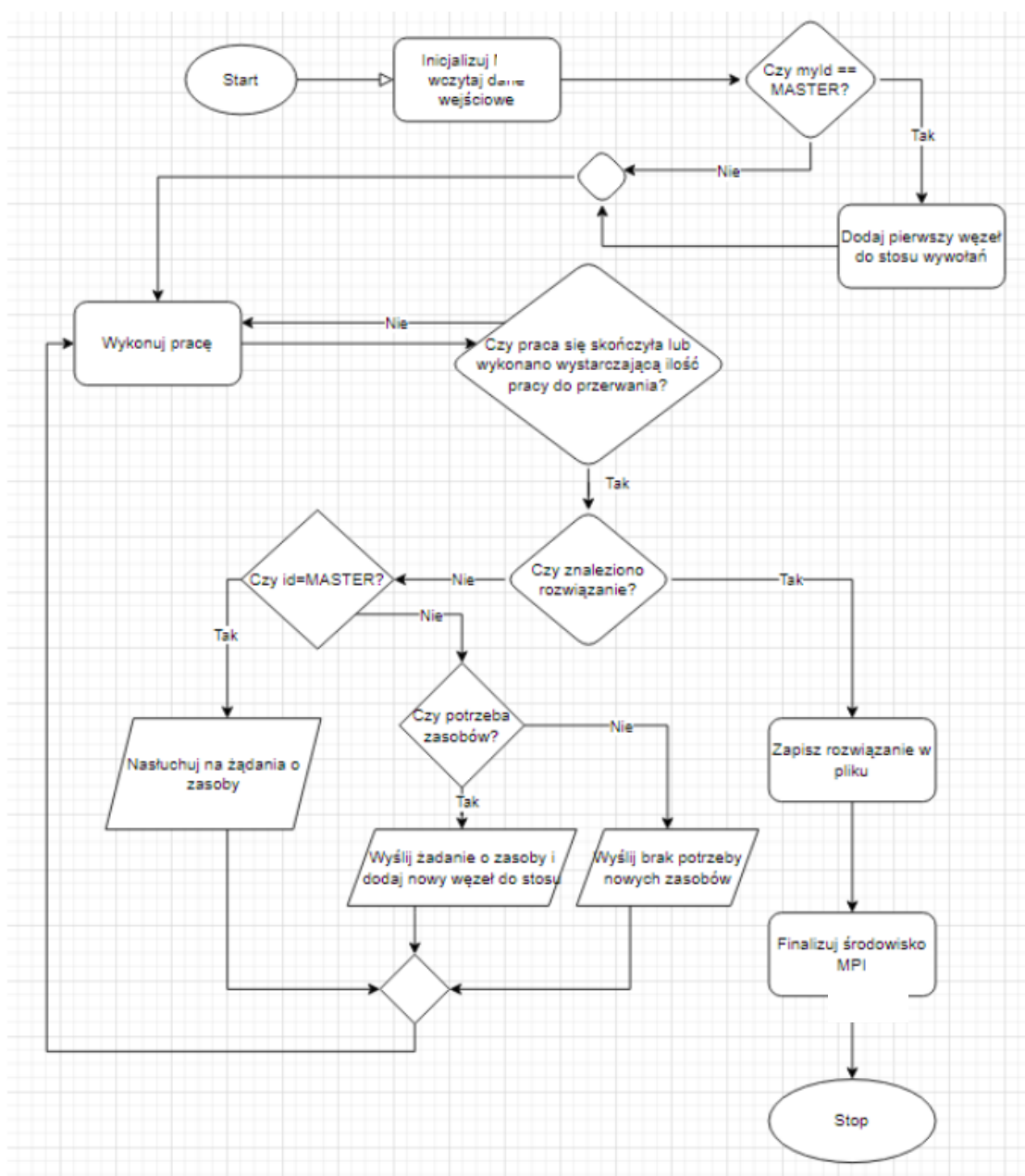
### 4. Budowa programu

Program opiera się o mechanizm Manager – Workers. Wszystkie procesy są zaangażowane w poszukiwanie rozwiązania, jednak proces o id=0 posiada dodatkową rolę rozdzielania zadań. Zadaniem workerów jest przeszukiwanie w głąb drzewa kolejnych sąsiednich pozycji w celu znalezienia ustawienia odpowiadającego pozycji końcowej. Na początku wykonania programu, pozycja wyjściowa zostaje wczytana z pliku „input.txt”. Następnie proces manager rozpoczyna przeszukiwanie w głąb od pozycji startowej.

W programie została zaimplementowana mechanika „dynamic load balancing” – jeśli procesorowi skończy się materiał do pracowania, to jest to wyłapywane przez menedżera, który przypisuje wątkowi nowy nieodwiedzony węzeł do przeszukiwania.

Jest to wykonywane przy pomocy wskaźnika globalnego, z którego każdy wątek jest w stanie pobrać kolejne przydzielone węzły do przeszukiwania. Po pewnej liczbie odwiedzonych węzłów, pracownicy wysyłają do managera informację czy dalej pracują (czy ich stos przeszukiwania nie jest pusty). Menedżer na bieżąco sprawdza czy pozostałe wątki pracują, w celu ewentualnego zbalansowania ich pracy.

. W przypadku znalezienia przez jakiś proces rozwiązania przeszukiwanie jest przerywane, a status rozpoznawany przez mastera. Master następnie zczytuje rozwiązanie od procesu, który zakończył przeszukiwanie sukcesem.



Rys 3. Schemat blokowy programu

## 5. Wykorzystane technologie.

Program został napisany w języku C++ w standardzie C++17 . Wersja środowiska UPC++ to 2022.3.0.

## 6. Obsługa programu.

Dane wejściowe znajdują się w pliku input.txt, natomiast dane wyjściowe zostają zapisane w pliku output.txt.

W celu obsługi programu, dodany został plik makefile udostępniający następujące polecenia: (przed wykonaniem poleceń, należy skonfigurować środowisko UPC++ przy pomocy: *source /opt/nfs/config/source\_upcxx.sh*)

- make (Skompilowanie programu źródłowego do postaci wykonywalnej na komputerach pracowni)
- make nodes (Polecenie tworzące plik nodes, zawierający listę działających komputerów w pracowni, wykorzystywany przy uruchomieniu programu)
- make out (Uruchomienie programu z danymi przykładowymi)
- make clean (Przywrócenie zawartości podkatalogu do stanu wyjściowego)