

Politechnika Wrocławska W12N



Politechnika Wrocławska W12N	
Prowadzący <i>Dr inż. Piotr Ciskowski</i>	Zadanie na ocenę: <i>5</i>
Nazwa przedmiotu <i>Projektowanie Algorytmów i Metody Sztucznej Inteligencji P</i>	Data wykonania ćwiczenia <i>30 marca 2022</i>
Imię, nazwisko, numer albumu <i>Wiktor Adasiak 258974</i>	

Spis treści

1	Cel ćwiczenia	2
1.1	Treść zadania	2
1.2	Założenia projektu	2
1.3	Środowisko pracy	2
2	Rozwój zadania	2
2.1	Idea projektu	2
2.2	Dobór struktury ADT	2
3	Sposób działania programu	3
4	Złożoność obliczeniowa	5
4.1	Funkcja sort_table	5
4.2	Złożoność obliczeniowa	6

1 Cel ćwiczenia

1.1 Treść zadania

Założmy, że Jan chce wysłać przez Internet wiadomość W do Anny. Z różnych powodów musi podzielić ją na n pakietów. Każdemu pakietowi nadaje kolejne numery i wysyła przez sieć. Komputer Anny po otrzymaniu przesłanych pakietów musi poskładać je w całą wiadomość, ponieważ mogą one przychodzić w losowej kolejności. Państwa zadaniem jest zaprojektowanie i zaimplementowanie odpowiedniego rozwiązania radzącego sobie z tym problemem. Należy wybrać i zaimplementować zgodnie z danym dla wybranej struktury ADT oraz przeanalizować czas działania - złożoność obliczeniową proponowanego rozwiązania. W sprawozdaniu (3-5 strony + strona tytułowa) należy opisać rodzaj wybranej struktury danych wraz z uzasadnieniem wyboru oraz opisać proponowane rozwiązanie problemu. Należy także opisać sposób analizy złożoności obliczeniowej i podać jej wynik w notacji dużego O .

1.2 Założenia projektu

Zadanie polegało na napisaniu programu, który miał przypominać w działaniu komunikator internetowy. Miał przede wszystkim posiadać funkcję podziału wiadomości na " n " części. Następnie program powinien posortować otrzymaną wiadomość w " n " częściach we właściwej kolejności, takiej samej jak wiadomość pierwotna.

1.3 Środowisko pracy

Program został przeze mnie napisany w języku C++. Program powstał na komputerze MacBook Pro M1 z wersją systemu macOS Monterey (12.2), natomiast edytor kodu jaki posłużył mi przy tym projekcie to Visual Studio Code w wersji 1.66.1. Program został skompilowany przy użyciu komendy: `"g++ -o app -Wall main.cpp -O0"`

2 Rozwój zadania

2.1 Idea projektu

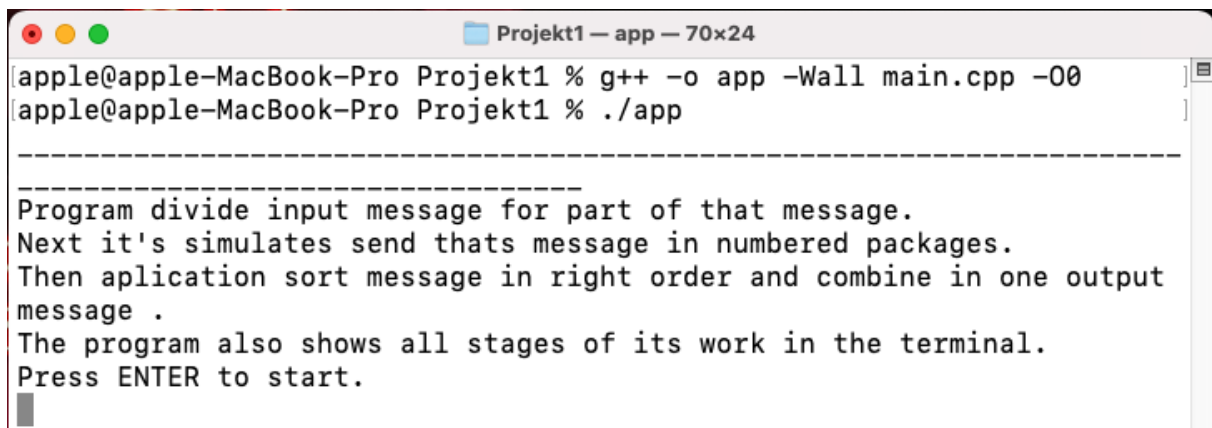
Celem działającego programu było przesyłanie wiadomości pomiędzy dwoma osobami. Wiadomość, która ma zostać poddana działaniu programu użytkownik podaje podczas pracy aplikacji. Po otrzymaniu wiadomości od użytkownika program musi podzielić ją na " n " pakietów. Jednakże po podzieleniu wiadomości, którą otrzymał program była ona w takiej samej kolejności jak wiadomość pierwotna, więc aby sprostac założeniom zadania postanowiłem stworzyć funkcję wysyłania pakietów w przypadkowym szyku. Kiedy taka paczka pakietów w losowej uporządkowaniu trafi do odbiorcy wiadomości pozostaje tylko ułożyć wiadomość w prawidłowym porządku.

2.2 Dobór struktury ADT

Mój wybór skierowałem ku kolejce priorytetowej na tablicy dynamicznej. Do tego typu zadania moim zdaniem jest to wybór, który od razu się nasuwa ze względu na to, że każdy pakiet posiada swój unikatowy numer dzięki, któremu sortowanie jest o wiele łatwiejsze. Implementacja tej struktury została prze mnie wykonana przy użyciu tablicy dynamicznej ze względu na brak informacji w momencie tworzenia programu o długości wiadomości wysyłanej przez nadawcę. Wykorzystanie takiej metody pozwala dobrać odpowiednią ilość pamięci, nie skończy nam się miejsce na wpisywanie naszej wiadomości (z zastrzeżeniem iż nie zostanie zajęta cała dostępna pamięć na urządzeniu, które obsługuje aplikację). Dodatkowo nie będziemy zajmować nadmiaru pamięci kiedy wiadomość wpisana przez użytkownika okaże się dość krótka.

3 Sposób działania programu

Początkowo program informuje nas o swoim działaniu:

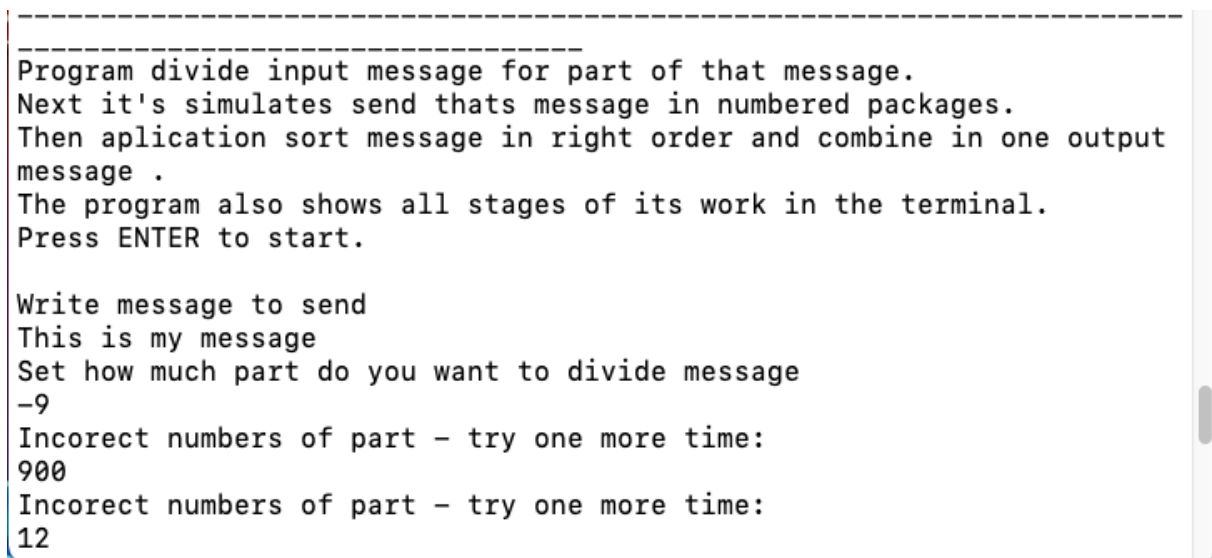


```
apple@apple-MacBook-Pro Projekt1 % g++ -o app -Wall main.cpp -O0
apple@apple-MacBook-Pro Projekt1 % ./app

-----
Program divide input message for part of that message.
Next it's simulates send thats message in numbered packages.
Then aplication sort message in right order and combine in one output
message .
The program also shows all stages of its work in the terminal.
Press ENTER to start.
█
```

Rysunek 1

Następnie prosi nas o wpisanie wiadomości, którą chcielibyśmy przesłać oraz na ile części chcemy ją podzielić. Warto zauważyć, że ilość części musi być odpowiednia tj. większa od 0 oraz mniejsza od długości wiadomości.



```
-----
Program divide input message for part of that message.
Next it's simulates send thats message in numbered packages.
Then aplication sort message in right order and combine in one output
message .
The program also shows all stages of its work in the terminal.
Press ENTER to start.

Write message to send
This is my message
Set how much part do you want to divide message
-9
Incorect numbers of part - try one more time:
900
Incorect numbers of part - try one more time:
12
```

Rysunek 2

Później program pokazuje jak podzielił wiadomość na dane pakiety, jak widać tutaj dalej są posegregowane

```
Table beffor shuffle
Th nr:1
is nr:2
 i nr:3
 s nr:4
my nr:5
 m nr:6
 e nr:7
 s nr:8
 s nr:9
 a nr:10
 g nr:11
 e nr:12
```

Rysunek 3

Następnie program wypisuje pakiety po symulacji przesyłu, czyli wypisuje pakiety w kolejności losowej

```
Table after shuffle
e nr:7
g nr:11
s nr:8
my nr:5
Th nr:1
s nr:9
a nr:10
e nr:12
is nr:2
 s nr:4
 i nr:3
 m nr:6
```

Rysunek 4

Na koniec program wypisuje już posortowane pakiety oraz wiadomość wyjściową, która jest scaleniem pakietów w odpowiedniej kolejności.

```
Table after sort
Th nr:1
is nr:2
 i nr:3
 s nr:4
my nr:5
 m nr:6
 e nr:7
 s nr:8
 s nr:9
 a nr:10
 g nr:11
 e nr:12
Output message:
This is my message
Enter to end%
apple@apple-MacBook-Pro Projekt1 %
```

Rysunek 5

4 Złożoność obliczeniowa

4.1 Funkcja sort_table

Poniżej przedstawiam zaimplementowaną prze mnie funkcję sort _ table:

```
////////// sort_table ////////////
// function get part of string i random order      //
// and size of table ,                             //
// next function with work loop for and while      //
// and using temporary table,                       //
// get right order which have original message,    //
// then that process is going to have all of start //
// array in temporary table                         //
// after all proces temporary table is delete      //
//////////
void sort_table(message input_tab[], int size_of_tab){
    message * temp_tab = new message[size_of_tab];
    int message_index = 1;

    for (int i = 0; i < size_of_tab; i++){
        int k=0;
        while (input_tab[k].nr != message_index)
        {
            k++;
        }
        temp_tab[i]=input_tab[k];
        message_index++;
    }
    for (int i = 0; i < size_of_tab; i++)
    {
        input_tab[i]= temp_tab[i];
    }
    delete[] temp_tab;
}
```

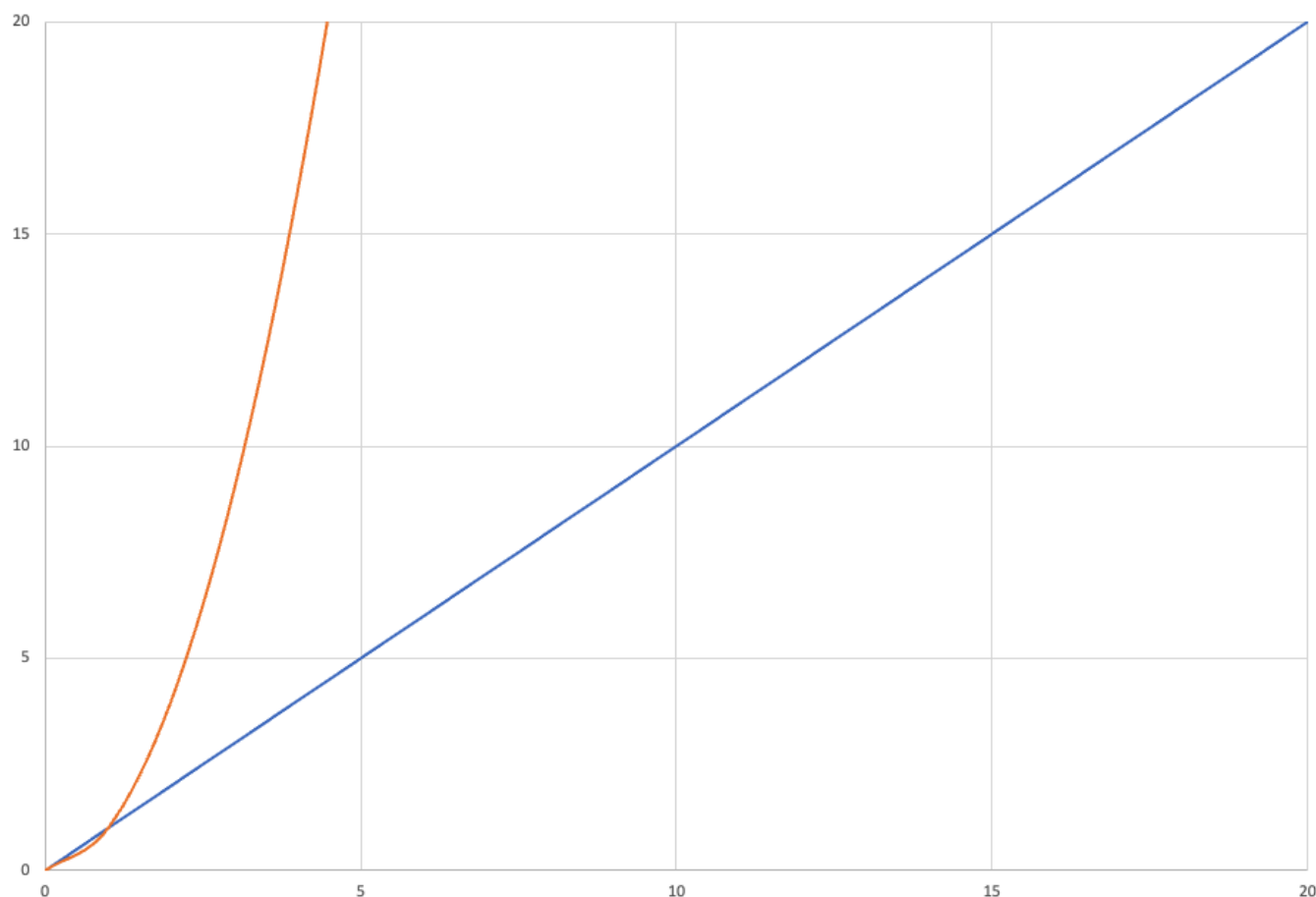
Rysunek 6

Funkcja przedstawia następujący sposób działania. Pobiera pakiety w losowej kolejności wraz z wielkością tablicy na , której została zapisana wiadomość. Następnie program przy użyciu funkcji "for" oraz "while" aplikacja dobiera używając tablicy pomocniczej odpowiednią kolejność jaka charakteryzowała wiadomość pierwotnie. Kolejno funkcja przypisuje elementy do tablicy pomocniczej a ta czynność powtarza się do momentu gdy nie wykona się dla wszystkich elementów tablicy pierwotnej. Z racji charakterystyki funkcji void należy przypisać wartości tablicy wejściowej ułożone już w odpowiedniej kolejności z pomocą tablicy pomocniczej. Następnie tablica pomocnicza jest usuwana.

4.2 Złożoność obliczeniowa

Główną część złożoności obliczeniowej tego programu generuje funkcja `sort_table`, reszta programu posiada złożoność obliczeniową liniową. Również funkcja `random_shuffle` jak podaje dokumentacja biblioteki `"std"` z której funkcja pochodzi ta złożoność liniową. W możliwie najbardziej korzystnej dla nas możliwości, czyli wtedy gdy pakiety na tablicy ułożone są w kolejności rosnącej wtedy funkcja przyjmuje wartości złożoności obliczeniowej równej $O(n)$, biorąc pod uwagę, że zmienna `"k"`, odpowiada ona za znajdowanie pasującego miejsca dla pakietu gdzie będzie pasował za pierwszym razem, , przez co `"while"` się nie wykona. Jednakże zakładając najgorszy przypadek, gdy elementy listy są ułożone w porządku rosnącym. Wtedy zmienna `'k'` zmuszona będzie przeszukać tabele `'n'` razy dla pierwszego przejścia `'for'`, `'n-1'` dla drugiego itd. Analizując pętlę `'for'`, która wykona się `'n'` razy oraz zmienna `'k'` w pętli `'while'` dla każdego przejścia `"for"` wykona się `"n"`, `"n-1"`, `"n-2"`, ... razy. W takim przypadku złożoność obliczeniowa w takim przypadku wynosić będzie $O(n^2)$

Można więc założyć, że złożoność obliczeniowa programu wynosi $O(n^2)$



Rysunek 7: Wykres niebieski $O(n)$, Wykres pomarańczowy $O(n^2)$