Paweł Świątkowski 293162

AAL - dokumentacja wstępna

Temat 5: Zalewanie wykresu słupkowego

Treść zadania:

Mając do dyspozycji ciąg nieujemnych liczb a1, a2 ... an reprezentujących kolejne słupki wykresu słupkowego Twoim zadaniem jest znalezienie największego obszaru ograniczonego dwoma słupkami, oraz osią x, który można zalać z góry. Odległości pomiędzy słupkami w osi x wynosi 1.
Uwaga, wody nie można wlewać "na" słupek (tak żeby rozlewała się na prawo i na lewo od niego).
Przykładowe rozwiązanie:
Wejście: N = [1,8,6,2,5,4,8,3,7]
Wyjście: 40
Wyjaśnienie: Najwięcej wody znajdzie się pomiędzy drugim, a 7 słupkiem o wysokości 8

Przeanalizowanie problemu:

Algorytm przyjmuje na wejściu ciąg n nieujemnych liczb reprezentujących wysokości słupków. Moim zadaniem jest znalezienie maksymalnej ilości wody jaka się zmieści pomiędzy dwoma dowolnymi słupkami. Założeniem zadania jest to, że wody nie wolno nalewać na słupek, w taki sposób że rozlewa się ona na jego lewo i prawo. Zostało również założone, że przed pierwszym i za ostatnim słupkiem nie ma niczego, czyli rozlana tam woda przepada.

Propozycja rozwiązania, algorytm optymalny:

Korzystamy z algorytmu gąsiennicowego. Moja gąsienica ma głowę i ogon, oba są iteratorami w liście reprezentującej wysokości kolejnych słupków. Poruszamy się raz od lewej strony ciągu oraz drugi raz od prawej strony.
W zadanym przejściu zaczynamy od początku. Przesuwamy głowę gąsienicy do czasu, gdy głowa będzie wyższa lub równa od wysokości ogonu gąsienicy. Wtedy obliczamy jaka ilość wody zmieści się w tej „dolinie” miedzy głową a ogonem gąsienicy (bierzemy pod uwagę odległość miedzy tymi iteratorami oraz wysokość mniejszej wysokości spośród wysokości ogona i głowy). Jeżeli znaleziona teraz pojemność jest większa niż dotychczasowa największa, bierzemy aktualną jako MaxCapacity.
Po przejściu ciągu od lewej oraz od prawej jako wynik algorytmu zwracamy największą znalezioną pojemność z obu przejść.
To rozwiązanie ma złożoność liniową.

Algorytm brutalny:

Pierwszym rozwiązaniem jakie stworzyłem było algorytmem brutalnym. Polegało ono na przechodzeniu od słupka i-tego do ostatniego słupka. W danej iteracji i przechodzę po kolejnych słupkach do czasu gdy napotkam słupek równy bądź wyższy od słupka i-tego. Obliczana jest wtedy odległość między tymi słupkami i mnożona jest przez minimum

wysokości i-tego słupka oraz tego słupka do którego dotarliśmy. Wykonując ten algorytm dla każdego i=1…n znajduję maksymalną pojemność.

Rozwiązanie algorytmem brutalnym ma złożoność kwadratową.

Jednak podczas testów pomiary czasu wykonania wskazują na złożoność liniową. Powodem tego jest to, że złożoność kwadratowa jest złożonością pesymistyczną, występującą tylko dla przypadku gdy dolina zajmuje znaczą część pola tzn. jest niewielka ilość dolin, a każda o dużej szerokości. W przypadkach testowych zwykle wychodziło wiele dolin o niewielkiej szerokości każda. Dlatego też średnio złożoność tego rozwiązania wychodzi liniowa, jednak z o wiele większym współczynnikiem (nachylenie prostej jest znacznie bardziej pionowe) niż w przypadku algorytmu optymalnego/

Sposób testowania poprawności rozwiązania:

Jakość algorytmu sprawdzam przez generowanie losowych „planszy" dla których wyznaczane jest rozwiązanie najpierw za pomocą algorytmu brutalnego, a następnie za pomocą mojego algorytmu optymalnego. Porównywane są wyniki, w ten sposób weryfikowana jest poprawność mojego rozwiązania.

Przeprowadzane były testy jakości, poprzez generowanie 10^{12} plan i dla każdego wygenerowanego przypadku oba rozwiązania były równe.

Testowanie wydajności:

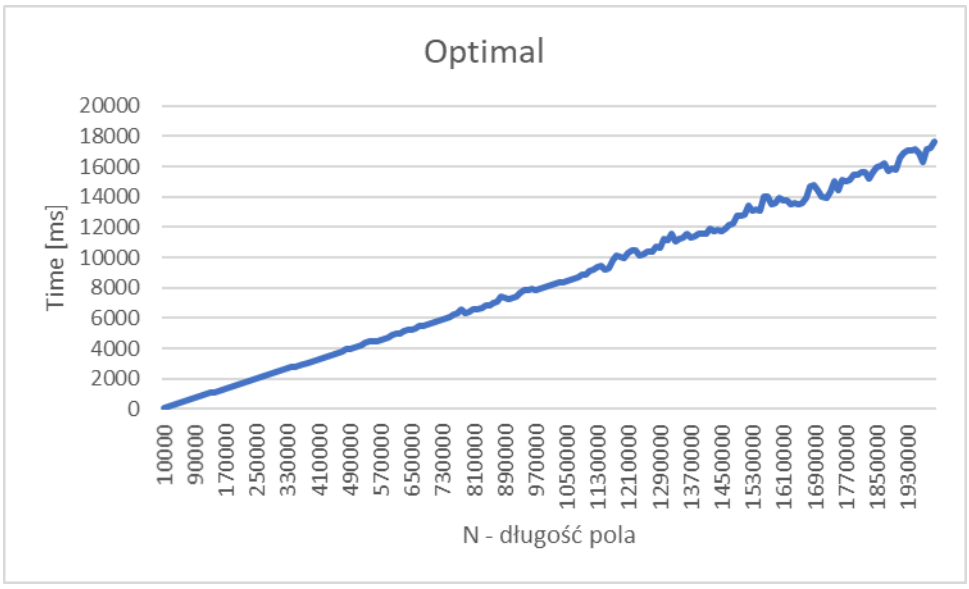
Złożoność obliczeniowa jest zależna od długości pola, czyli ilości słupków na wykresie słupkowym w którym to właśnie poszukujemy maksymalnej ilości wody jaka się zmieści.

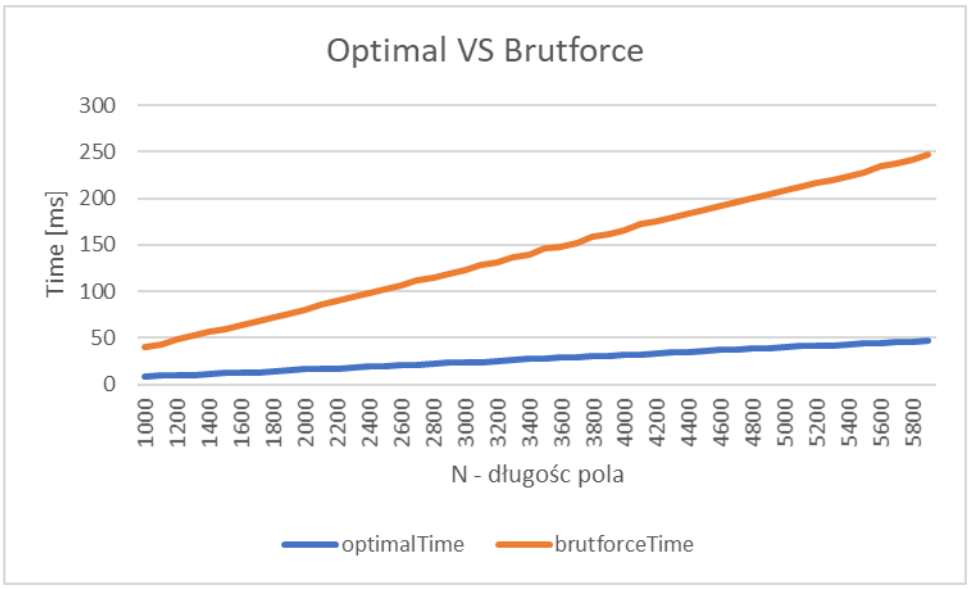
Dla zadanej wielkości problemu (długości planszy) obliczenia wykonywane są wielokrotnie po czym wyciągną jest wartość średnia czasu obliczania rozwiązania, aby zredukować wpływ łatwych lub trudnych wygenerowanych przypadków. Interesuje nas złożoność zamortyzowana algorytmu.

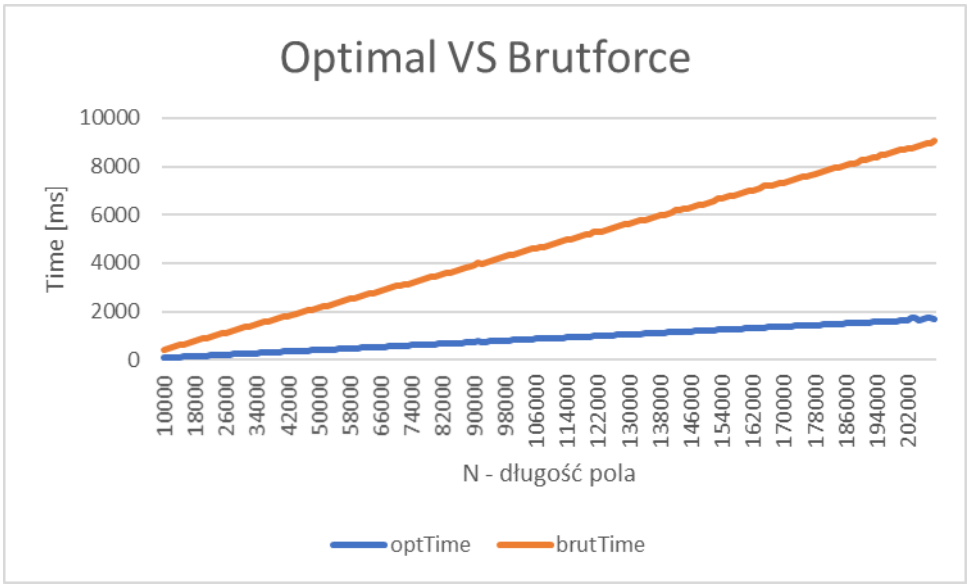
Do testowania wydajności obu algorytmów potrzebne jest podanie kilku parametrów:

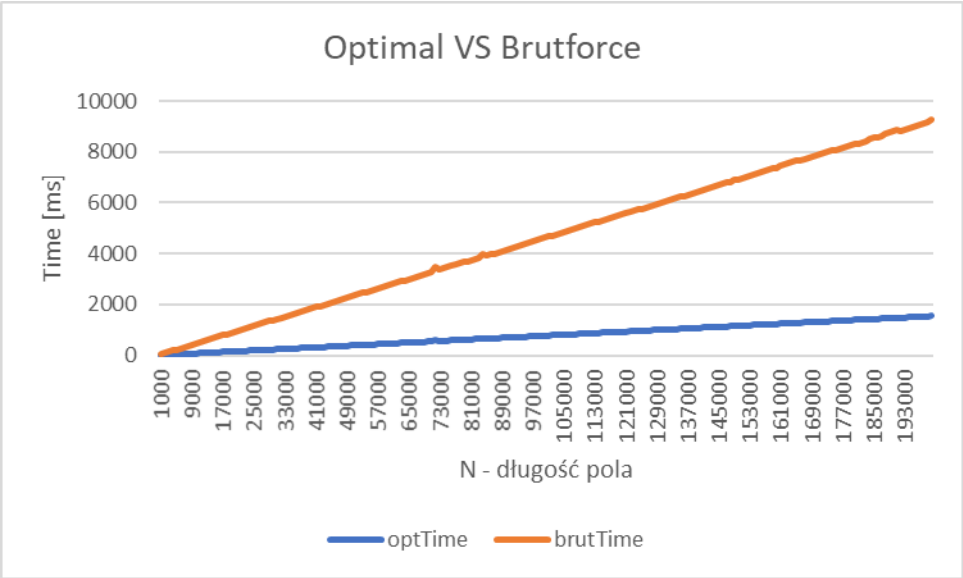
- offset – oznacza długość pola od jakiej zaczną się obliczenia
- maksymalna wysokość słupków - jest to parametr potrzebny dla generatora plansz
- wielkość kroku – o tyle będzie się zwiększała długość pola
- ilość kroków – ile iteracji wartości długości pola (offset + wielkoscKroku*iteracja) zostanie wykonanych
- ilość próbek pomiarowych na krok – ile pomiarów czasu będzie wykonywanych dla zadanej wartości długości pola

Przykładowe wyniki przeprowadzonych pomiarów:


Optimal
Time [ms]
20000
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
10000
90000
170000
250000
330000
410000
490000
570000
650000
730000
810000
890000
970000
1050000
1130000
1210000
1290000
1370000
1450000
1530000
1610000
1690000
1770000
1850000
1930000
N - długość pola


Optimal VS Brutforce
Time [ms]
300
250
200
150
100
50
0
1000
1200
1400
1600
1800
2000
2200
2400
2600
2800
3000
3200
3400
3600
3800
4000
4200
4400
4600
4800
5000
5200
5400
5600
5800
N - długosc pola
optimalTime
brutforceTime


Optimal VS Brutforce
Time [ms]
10000
8000
6000
4000
2000
0
10000
18000
26000
34000
42000
50000
58000
66000
74000
82000
90000
98000
106000
114000
122000
130000
138000
146000
154000
162000
170000
178000
186000
194000
202000
N - długość pola
optTime
brutTime



Tryby uruchomienia:

1. **Tryb interaktywny** - po uruchomieniu programu zostaje wyświetlone menu główne
    a. <u>Pojedyncze rozwiązanie</u>
    Obliczenie maksymalnej ilości wody jaka się zmieści w podanym przypadku
    Argumenty: [dlugosc pola], [wysokości słupków (ich ilość ma się równać podanej długości pola)]
    b. <u>Generowanie przypadków testowych i porównanie wyników dla algorytmu Optimal oraz Brutforce</u>
    Generowane są przypadki testowe o zadanych parametrach, a następnie są one rozwiązywane za pomocą algorytmu Optimal oraz Brutforce. Porównywane są jakie zwracają te algorytmy. Jeżeli zostanie wygenerowana próba w której te dwa algorytmy zwrócą różne wartości wyświetlony zostanie błąd.
    Argumenty: [dlugosc pola], [max wysokosc slupka], [ilosc prob]
    c. <u>Pomiar czasu wykonania dla coraz większych próbek dla algorytmu Optimal</u>
    Wykonanie pomiarów wydajnościowych algorytmu Optimal. Czas wykonania zależny jest od długości pola, z tego powodu to właśnie ta wartość jest zwiększana w kolejnych krokach testowych. Dla danej długości pola generowanych jest wiele przypadków testowych i wykonywane są dla nich pomiary czasu rozwiązania, następnie czas wykonania zostaje uśredniony.
    Argumenty: [offset - od jakiej długości mają zaczynać się kroki pomiaru], [max wysokosc slupka], [wielkosc kroku], [ilosc krokow], [ilosc probek pomiarowych na dany krok]
    d. <u>Pomiar czasu wykonania dla coraz większych próbek dla obu algorytmów</u>
    Analogicznie do poprzedniego punktu, tylko w tym przypadku wygenerowane próbek rozwiązywane są za pomocą algorytmu Optimal oraz Brutforce. Wyniki pomiarów są zapisywane a następnie wyświetlane.

Argumenty: [offset - od jakiej długości mają zaczynać się kroki pomiaru], [max wysokosc slupka], [wielkosc kroku], [ilosc krokow], [ilosc probek pomiarowych na dany krok]

2. **Tryb wsadowy**

   Pierwszym argumentem jest operacja jaka ma zostać wykonana (dostępne operacje są takie same jak w trybie interaktywnym):

   1 - Pojedyncze rozwiązanie

   2 - Generowanie przypadków testowych i porównanie wyników dla algorytmu Optimal oraz Brutforce

   3 - Pomiar czasu wykonania dla coraz większych próbek dla algorytmu Optima

   4 - Pomiar czasu wykonania dla coraz większych próbek dla obu algorytmów

   Każda z operacji przyjmuje argumenty analogiczne do tych opisanych w trybie interaktywnym