Dokumentacja projektu

Autor: Agnieszka Felis

Tytuł: Algorytm Kadane dla tablic 1D i 2D.

1. Zarys problemu.

Algorytm Kadane, znany również jako problem maksymalnej sumy podtablicy polega na znalezieniu ciągłej podtablicy o największej możliwej sumie w danej jednowymiarowej lub wielowymiarowej tablicy liczb. Liczby w tablicy wejściowej mogą być zerem lub być dodatnie, ujemne.

Np. dla tablicy `[-2,1,-3,4,-1,2,1,-5,4]` ciągła podtablica to `[4,-1,2,1]` z sumą `6`.

Właściwości problemu:

- Jeżeli w tablicy wszystkie liczby są dodatnie to problem jest trywialny – maksymalna podtablica to cała tablica.
- Jeżeli w tablicy wszystkie liczby są ujemne to rozwiązaniem jest pusta tablica, zatem suma to 0.
- Może zdarzyć się tak, że kilka różnych podtablic posiada tą samą maksymalną sumę.

Algorytm dla tablic jednowymiarowych działa w czasie O(n) a dla dwuwymiarowych w czasie $O(n^3)$

2. Zarys działania programu.

Program przy użyciu funkcji `kadane_for_1D(matrix)` i `kadane_for_2D(matrix)` odpowiednio dla tablic jednowymiarowych i dwuwymiarowych (kwadratowych i prostokątnych) znajduje największe ciągłe podtablice. Wypisuje sumę ich wartości oraz indeks początka i końca tablicy (lewego górnego rogu i prawego dolnego rogu).

Przykład dla tablicy jednowymiarowej:

```
A = [-2,3,-1,2]
```

Gdy zastosujemy funkcję `kadane_for_1D(A)` otrzymamy wynik `(4,1,3)`:

- `4` = suma największej podtablicy
- `1` = indeks pierwszego elementu należącego do niej
- `3` = indeks ostatniego elementu należącego do niej

Tak więc, największą podtablicą dla naszej tablicy jednowymiarowej A jest `[3,-1,2]`

Przykład dla tablicy dwuwymiarowej:

```
B = [[1,1,1],[2,2,2]]
```

Gdy zastosujemy funkcję `kadane_for_2D(B)` otrzymamy wynik `(9, [0,0], [1,2])`:

- `9` = suma największej podtablicy
- `[0,0]` = pierwszy róg tablicy
- `[1,2]` = drugi róg tablicy

3. Zawartość katalogu

Katalog zawiera 3 moduły:

- `dokumentacja.pdf` – zawierająca opis działania programu
- `main.py` – implementujący algorytm Kadane dla jednowymiarowej i wielowymiarowej tablicy
- `test.py` – zawierający testy funkcji napisane przy użyciu biblioteki `Pytest`

4. Sposób uruchomienia programu

- Aby wyświetlić wyniki testów do programu należy umieścic plik `main.py` i `test.py` w jednym folderze a następnie skompilować i uruchomić program `test.py`
- Aby dodać własne przykłady do algorytmu w pliku `main.py` należy skorzystać z tablicy dostępnych w bibiliotece `numpy` oraz wpisać wartości. Następnie taką tablicę przekazujemy jako argument do wybranej funkcji `kadane1D()` lub `kadane2D()` a wynik wypisać na ekranie przy użyciu funckji `print()`.

Przykład:

```
A = np.array([5,7,-1,3,0,9,7,-3])

print(kadane1D(A))
```

5. Opis działania programu.

Obie funkcje zawarte w programie mają za zadanie znaleźć ciągłą podtablicę o największej możliwej sumie.

Funkcja `kadane_for_1D` dla tablic jednowymiarowych przy każdym kroku dodaje do aktualnej sumy (`sum_current`) wartość znajdującą się pod obecnie sprawdzanym indeksem. Jeżeli suma ta jest większa od maksymalnej sumy (`sum_max`) to zapamiętuje indeksy startu i uaktualnia sumę maksymalną (obecną sumą). W innym przypadku jeżeli suma jest mniejsza od 0, zeruje sumę i zmienia indeks startu na kolejny. Gdy dochodzi do końca tablicy wypisuje zapisane wartości największej sumy, indeks startu i indeks końca poddtablicy. Jeśli w podanej tablicy będzie żadnego dodatniego elemnetu to funkcja zwróci: `[0,-1,-1]`.

Funkcja `kadane_for_2D` wykorzystuje funkcję `kadane_for_1D` stosując ją wiele razy dla sum poszczególnych kolumn, np. gdy mamy macierz złożoną z 3 kolumn:

- Najpierw `kadane_for_1D()` zostanie zastosowane dla kolumny nr 1
- Dla sumy kolumny nr 1 i nr 2
- Dla sumy kolumny nr 1, nr 2 i nr 3
- Natępnie `kadane_for_1D()` zostanie zastosowane dla kolumny nr 2
- Dla sumy kolumny nr 2 i nr 3
- Finalnie `kadane_for_1D()` zostanie zastosowane dla ostatniej kolumny – kolumny nr 3

Jeśli w podanej tablicy będzie żadnego dodatniego elemnetu to funkcja zwróci: `[0, [-1,-1], [-1, -1]]`.

Dzięki wykorzystaniu funkcji `kadane_for_1D`, której złożoność wynosi `O(n)` złożoność kolejnej funkcji `kadane_for_2D` wynosi $O(n^3)$, ponieważ wywołuje ją $\sim n^2$ razy.