Tomasz Załuska

AAL projekt – temat: gra na giełdzie

1. Opis problemu.

Zakładamy że posiadamy idealny algorytm oparty o zaawansowane metody uczenia maszynowego, który bezbłędnie przewiduje ceny akcji firmy XYZ w kolejnych dniach. Znając te predykcje (wpisane w liście). Zadaniem algorytmu jest opracowanie optymalnej sekwencji kupna i sprzedaży akcji w ciągu najbliższych n dni. W ciągu jednego dnia można kupić jedną akcję spółki XYZ, sprzedać dowolną ilość akcji, lub czekać na lepszą okazję. Jako wyjście programu, zwróć całkowity profit który można uzyskać.

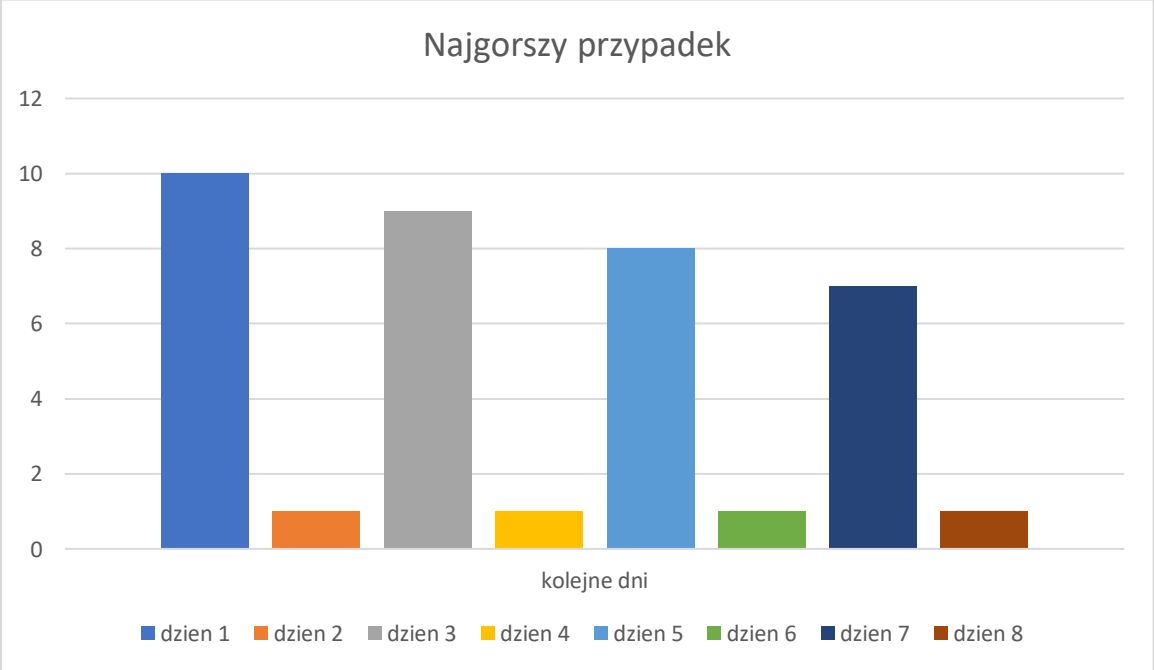
2. Rozwiązanie.

Dla podanej tablicy cen szukamy maksimum specjalna funkcją max_plus() i następnie kupujemy wszystkie akcje aż dojdziemy do tego dnia, w którym cena akcji jest maksymalna. W tym dniu sprzedajemy wszystkie nabyte akcje. Z pozostałych dni szukamy nowego maksimum. I tak w kółko aż przejrzymy cała tablice z cenami akcji.

Opis funkcji max_plus().

Funkcja ta szuka maksimum tablicy w klasyczny sposób, jednak gdy początkowa wartość tablicy jest maksimum oraz kolejne wartości tablicy są malejące, funkcja ta odcina wartości od początku tablicy do momentu gdy wartości tablicy zaczynają rosnąć.

3. Obliczenie złożoności.

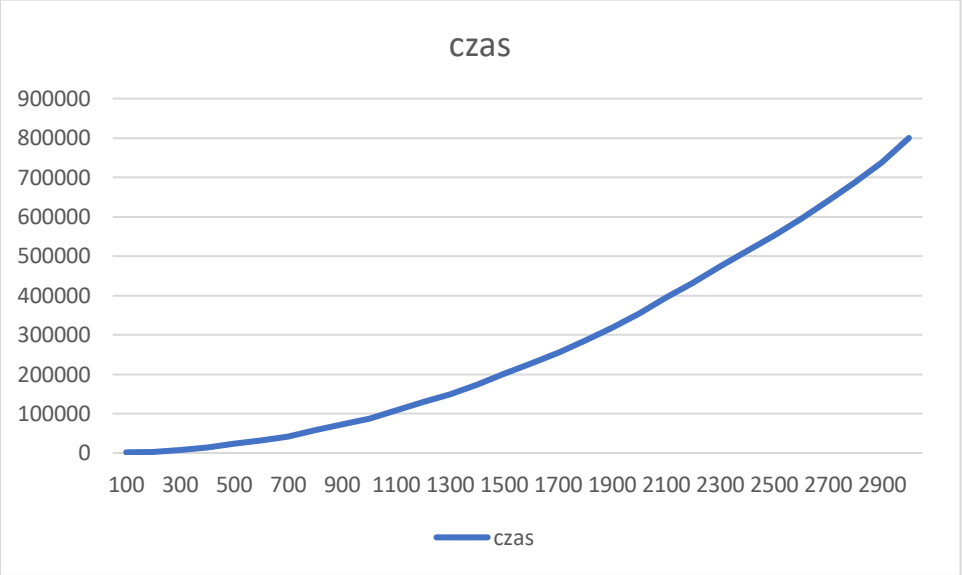


Najgorszy przypadek dla opisanego algorytmu to taki, w którym wartość akcji jest malejąca co drugi dzień, a w pozostałe dni wartości jest minimalna. W tym przypadku opisany algorytm odcina najmniejsza możliwą część tablicy wartości akcji.

Złożoność obliczeniowa wynosi: $\sum_{i=0}^{\frac{n}{2}-1}(n-2i)+(n-2i-2)+2=\sum_{i=0}^{\frac{n}{2}-1}2n-4i=\frac{1}{2}n(n+2)$ = O(n^2)

Złożoność pamięciowa: O(n)

4. Wykres złożoności wynikająca z testowania.



Podany czas w nanosekundach.

Jak widać na wykresie z testów wynika, że złożoność podanego algorytmu jest kwadratowa co zgadza się z obliczeniami teoretycznymi.

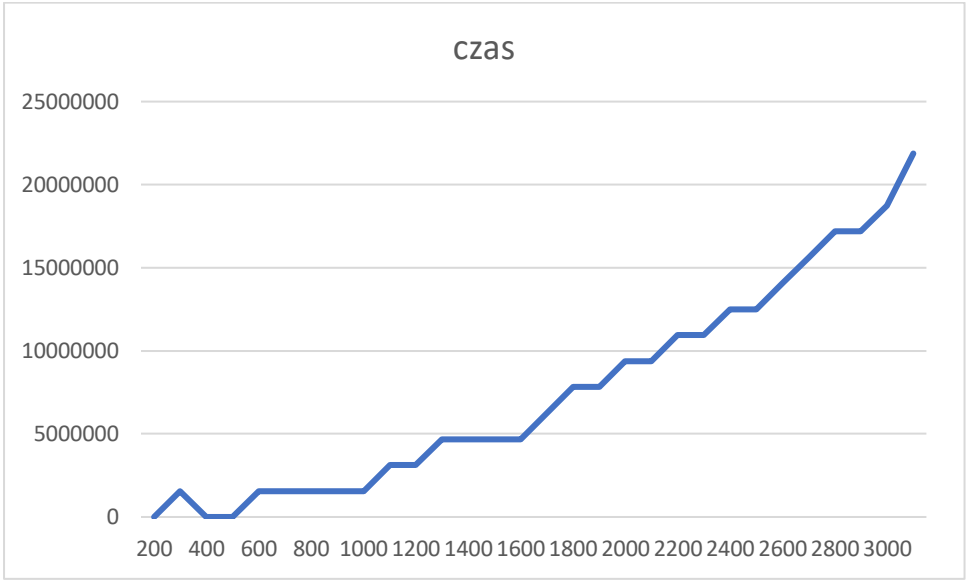
Porównanie złożoności wynikającej z testów , z obliczeniami teoretycznymi.

N	t(n) [ns]	q(n)
200	3125000.0	0.8521410500671518
300	9375000.0	1.1399502788977791
400	14062500.0	0.9634281275012947
500	23437500.0	1.0286802317543704
600	31250000.0	0.9531145742722296
700	43750000.0	0.9808119208750122
800	57812500.0	0.9926593217272276
900	73437500.0	0.9965788530842004
1000	90625000.0	0.9963757028529691
1100	109375000.0	0.9940005319212609
1200	131250000.0	1.0024354781987959
1300	154687500.0	1.0068011768428986
1400	179687500.0	1.0085203073700801
1500	206250000.0	1.0084992880155585
1600	232812500.0	1.0006141278449256
1700	262500000.0	0.9994559575267458
1800	301562500.0	1.0242191077172889
1900	331250000.0	1.0097982270207457
2000	368750000.0	1.0145671363336979

5. Wnioski.

Opisany algorytm dzięki swojemu usprawnieniu funkcji szukającej wartości maksymalnej działa nieco lepiej ,niż gdyby zamiast funkcji max_plus() zastosować zwyczajną funkcje szukają maksymalnej wartości. Algorytm z zastosowaniem tradycyjnej funkcji max() miał by złożoność wynoszącą n^2, więc jak widać dzięki zastosowaniu ulepszonej funkcji zmniejszył się parametr przy n^2 z 1 do ½.

Wykres gdy zastosowano zwykłą funkcje max()



Jak widać algorytm z funkcja max_plus() działa znaczniej lepiej.

Oba rozwiązania były również testowane na tablicach o całkowicie losowych wartości cen akcji.

W tych przypadkach również algorytm z funkcją max_plus() działa szybciej.