Algorytm pakujący kartony

Kacper Biegajski, AAL 19Z

➔ Problem

Ortodoksyjny kolekcjoner kartonów zaczyna narzekać na brak miejsca. Postanowił oszczędzić miejsce przez wkładanie kartonów jeden w drugi. W trosce o zachowanie dobrego stanu kartonów wkłada tylko jeden karton wewnątrz większego, a wolną przestrzeń wypełnia materiałem ochronnym. Tak zabezpieczony karton umieszcza wewnątrz innego większego kartonu. Nie może schować 2 kartonów obok siebie w innym kartonie. Dla danego zbioru kartonów znaleźć najlepsze upakowanie, czyli zwalniające najwięcej miejsca.

➔ Algorytm

Celem jest minimalizacja sumy objętości kartonów będących na zewnątrz.

- Sortujemy kartony po ich objętości od największego do najmniejszego.
- Bierzemy największy dostępny karton, traktujemy go jako aktywny, wkładamy jako pierwszy element stosu i usuwamy z listy kartonów.
- Przechodzimy na następny w kolejności kartonu.
- Jeżeli dany karton mieści się w aktywnym kartonie, czyli ma wszystkie krawędzie mniejsze, to wkładamy go jako kolejny element stosu, traktujemy jako aktywny i usuwamy z listy kartonów.
- Powtarzamy punkty 3 i 4 aż dojdziemy do końca tablicy.
- Zamykamy stos.
- Jeżeli lista kartonów nie jest pusta to wracamy do pkt. 2.
- Zapisane stosy tworzą optymalne upakowanie kartonów.

Jest to algorytm zachłanny, jednak w tym problemie jest to algorytm dający bardzo dobre wyniki. Nie znalazłem żadnego kontrprzykładu, w którym algorytm ten dawałby gorszy wynik od optymalnego. Nawet jeśli istnieją takie przykłady, to rezultat działania algorytmu jest akceptowalny (przeważnie dobre wyniki w rozsądnym czasie).

➔ Szczegóły implementacyjne

- język: Python wersja 3.7
- użyte struktury danych
  - list (zbiór wszystkich kartonów, stosy upakowanych kartonów)
  - tuple (reprezentacja kartonu (x, y, z, V))
- algorytmy pomocnicze
  - Timsort (wykorzystywany w pythonowej funkcji sort())

➔ Złożoność

- spodziewana T(n) = $O(n^2)$
  Algorytm ma dwie pętle (jedna zagnieżdżona w drugiej), które w pesymistycznym wypadku wykonują po n iteracji (dla każdego kartonu będziemy musieli sprawdzić czy pozostałe mniejsze kartony zmieszczą się do środka)
- wyniki pomiarów czasowych (3 tryb)

$$q(n) = \frac{t(n)\ T(n_{\ mediana})}{t(n_{\ mediana})\ T(n)}$$

n	t(n) [s]	q(n)
1000	0.23313040733337403	1.0145771678283924
2000	0.9089640617370606	0.9889466950746395
3000	2.7007328987121584	1.3059462044720316
4000	3.587327241897583	0.9757468885090217
5000	5.380461263656616	0.9366248209200687
6000	7.8987349510192875	0.9548633015813103
7000	10.618611383438111	0.9431001394533345
8000	14.129860830307006	0.9608245087883721
9000	18.397417736053466	0.9884575272118111
10000 (mediana)	22.978085327148438	1.0
11000	26.560509347915648	0.9552943101789412
12000	31.368974876403808	0.9480341821647921
13000	38.317931985855104	0.986737241732694
14000	50.054562997817996	1.1114086305377455
15000	53.544173908233645	1.0356568133100783
16000	61.122921562194826	1.039083147934126
17000	68.02258052825928	1.0243336100973754
18000	73.70878634452819	0.9900574869960693
19000	88.10103387832642	1.062086687310635

Test był przeprowadzany dla danych trudnych (pesymistycznych). q(n) dla wszystkich n oscyluje w okolicach 1, co oznacza że spodziewana złożoność została dobrze oszacowana.