Filip Przybysz 293137

Zadanie 2 – Problem Plecakowy

Dokumentacja

1. Zadanie

Do walizki o ograniczonej pojemności C chcemy załadować przedmioty o jak największej wartości, mając jednak na uwadze, że każdy z nich zajmuje pewną objętość. Mając n przedmiotów wraz z n-elementową tablicą odpowiadających im wartości $\{p_i\}$ oraz objętości $\{c_i\}$, znajdź zestaw rzeczy mieszczących się w walizce o największej sumarycznej wartości. *Uwaga:* Możemy dobierać maksymalnie m przedmiotów tego samego typu (tzn. o tej samej wartości i tej samej objętości).

2. Propozycja rozwiązania

Wybrany język programowania: C++

Zakładam, że wartości C, n, m oraz wszystkie c_i należą do liczb naturalnych dodatnich a wartości p_i do liczb rzeczywistych.

Zadanie sprowadza się do ograniczonego problemu plecakowego, gdzie maksymalna ilość wziętych sztuk każdego przedmiotu jest ograniczona przez tę samą stałą – m.

Zgodnie z zaleceniami dr inż. Łukasza Skoniecznego przedstawiam propozycje dwóch rozwiązań dla porównania. Jednego bardzo brutalnego a drugiego dynamicznego. Obydwa algorytmy gwarantują znalezienie rozwiązania optymalnego jednak różnią się złożonością.

2.1. Rozwiązanie brutalne

Przegląd zupełny przestrzeni wszystkich rozwiązań spełniających warunki zadania wymaga w pesymistycznym przypadku sprawdzenia $(m+1)^n$ kombinacji (n elementów, które mogą być wybrane $0..m$ razy) i wyboru najlepszej.

Algorytm polega na wygenerowaniu tych wszystkich kombinacji i zapamiętaniu najlepszego, dotychczas znalezionego i spełniającego warunki zadania, rozwiązania.

Dla operacji podstawowej porównania rozwiązań osiąga więc pesymistyczną złożoność $O((m+1)^n)$.

2.2. Rozwiązanie dynamiczne

Algorytm dynamiczny polega na budowaniu i zapamiętywaniu najlepszych rozwiązań $A(j)$ dla walizek o kolejnych pojemnościach $j = 0..C$.

Dla walizki o zerowej pojemności największa sumaryczna wartość przedmiotów mieszczących się w niej - $a(0) = 0$. Zapamiętujemy strukturę $A(0)$ rozwiązania, zawierającą wartość $a(0)$ oraz wyzerowaną n-elementową tablicę liczników pokazującą wykorzystanie poszczególnych przedmiotów.

Dla kolejnych wartości $j = 1..C$ budujemy najlepsze możliwe rozwiązanie. Na początku za najlepsze rozwiązanie $A_{max}(j)$ przyjmujemy $A(j-1)$. Następnie przeglądamy n-elementową tablicę przedmiotów. Jeżeli $c_i \leq j$ oraz rozwiązanie $A(j-c_i)$ nie wykorzystało jeszcze m elementów o indeksie i to porównujemy $a(j-c_i)+p_i$ z $a_{max}(j)$.
Jeżeli $a(j-c_i)+p_i$ jest większe to zastępujemy $A_{max}(j)$ rozwiązaniem powstałym z $A(j-c_i)$ poprzez inkrementację licznika wykorzystanych elementów o indeksie i oraz przypisanie $a_{max}(j) \coloneqq a(j-c_i)+p_i$.
Po sprawdzeniu wszystkich indeksów $i = 1..n$ przedmiotów z tablicy przypisujemy $A(j) \coloneqq A_{max}(j)$.

Jeżeli $j = C$ – zwracamy rozwiązanie $A(C)$ jako wynik działania algorytmu. W przeciwnym wypadku powtarzamy proces budowania rozwiązania dla walizki o pojemności $j+1$.

Jako że wymagane jest zbudowanie C rozwiązań a budowa każdego wymaga przejścia po n-elementowej tablicy przedmiotów, pesymistyczna złożoność algorytmu, dla operacji podstawowej porównania rozwiązań, wynosi $O(n * C)$.

3. Konwencja wejścia/wyjścia

Na wejściu pojawią się kolejno wartości C, m i n oraz n par wartości p_i c_i.

Na wyjściu pojawi się n trójek wartości p_i c_i m_i , gdzie m_i oznacza ilość przedmiotów o wartości p_i i objętości c_i wykorzystanych w znalezionym rozwiązaniu. Na samym końcu znajdzie się też liczba rzeczywista oznaczająca sumę wartości przedmiotów wykorzystanych w znalezionym rozwiązaniu.

4. Dane testowe

W testach wzorował się będę na schemacie przedstawionym przez Davida Pisingera w artykule "Where are the hard knapsack problems?"[1].

Wykorzystam zbiory losowo generowanych punktów z silnie skorelowanymi wartościami c_i i p_i. Objętości c_i losowane będą z rozkładem jednostajnym z zakresu $1..R$. Wartości obliczane będą wedle wzoru $p_i = c_i + R/10$.

Przetestowane zostaną przypadki dla: $n \in \{50, 100, 200, 500, 1000, 2000\}$, $m \in \{2, 3, 5\}$ oraz $R \in \{1000, 10000\}$.
Dla każdego przypadku wygenerowane zostanie $H = 100$ instancji a pojemność walizki w każdej instancji $h = 1..H$ będzie ustalana jako $C_h = \frac{h}{H+1} \sum_{i=1}^{n} c_i * m$.

Wynikiem testu dla danego przypadku będzie średni czas pracy algorytmu rozwiązującego problem.

[1] "Computers & Operations Research" Volume 32, Issue 9, September 2005

5. Wyniki testów

Początkowo zakładane wielkości problemów okazały się nazbyt optymistyczne. Testy wykonywały się godzinami więc znacznie ograniczono wartości n, m oraz H. Wartości czasu mierzone są tikami domyślnego zegara std::chrono::steady_clock.

Zmierzone wartości wskazują na dość dobre oszacowanie teoretyczne złożoności asymptotycznej.

Brutalny		Dynamiczny	
H = 5		H = 20	
n	q(n)	n	q(n)
		20	0.866762
6	0.956306	30	0.963737
8	0.994225	40	0.959901
10	1.04455	50	1.00947
12	1	60	1
14	1.0386	70	1.01071
16	0.989246	80	0.990495
18	1.0188	90	0.971825
		100	0.961141