Praca domowa 02 – knapsack

Szlachetka Miłosz
Nr albumu. 114014

1.Cel zadania

Celem zadania była implementacja programu w języku Java, który odczytuje z podanego pliku wymiary prostokątnych elementów a następnie dla podanego rozmiaru kwadratowej tafli (n x n) wyznacza najlepsze ułożenie elementów na owej tafli i wylicza ilość niewykorzystanej powierzchni.

2.Realizacja zadania

Program składa się z dwóch plików : Knapsack.java oraz Main.java.

Knapsack.java:

- atrybuty:
  - **private long regionFreeArea** - pozostałe, wolne miejsce na tafli.(Początkowo równe całkowitemu polu powierzchni tafli)
  - **private final Main.Region baseRegion** tafla, która będzie wypełniana prostokątnymi elementami. Struktura ta ma charakter drzewiasty.
- metody:
  - **int pack(List<Main.Rectangle> rectangles)** - odpowiada za obliczanie niewykorzystanej powierzchni na tafli. Dla każdego elementu znajduje odpowiednie miejsce na tafli, a następnie umieszcza go w odpowiedniej części tafli i zmniejsza jej rozmiar.
  - **private Main.Region findSuitableRegion(Main.Region region,Main.Rectangle rectangle) -** znajduje miejsce na tafli, które może pomieścić zadany, prostokątny element. Dopasowywanie odbywa się poprzez szukanie odpowiedniego kawałka tafli oraz obracanie elementu. Wykorzystuje metodę **checkFreeRegion**.
  - **private Main.Region checkFreeRegion(Main.Region region, Main.Rectangle rectangle)** - sprawdza **niezajęty** region i jeśli ma on odpowiedni rozmiar aby pomieścić dany prostokąt. Prostokąt jest obracany w razie potrzeby. Jeśli rozmiar jest za mały zwraca null.
  - **private void divideRegion(Main.Region region, Main.Rectangle rectangle) -** dokonuje podziału zadanego kawałka tafli na dwa mniejsze obszary (na prawo oraz poniżej aktualnego kawałka tafli zawierającego prostokątny element)

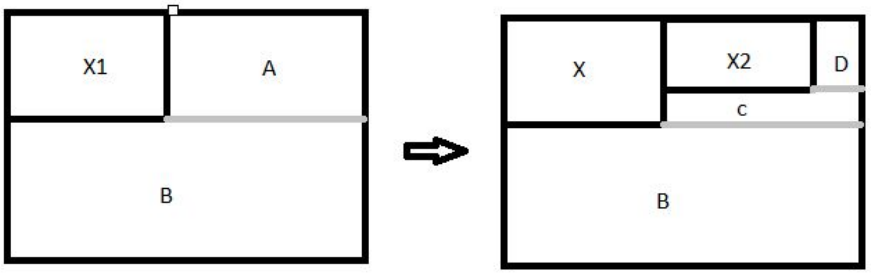
Main.java:

- statyczne klasy wewnętrzne:

    - **public static class Rectangle** - klasa, która reprezentuje prostokątne elementy umieszczane na tafli. Posiada takie atrybuty jak: długość, szerokość, pole powierzchni oraz region(obszar) w którym jest umieszczony. Posiada metodę **void rotate()**, która wykonuje obracanie elementu
    - **static class SortByAreaDescending implements Comparator<Rectangle> -** klasa implementująca metodę compare niezbędną do sortowania elementów typu Rectangle
    - **public static class Region** - klasa reprezentująca kawałek tafli. Posiada takie atrybuty jak: współrzędne lewego górnego rogu (x,y), szerokość, długość, zmienną boolean “rectanglePlaced”, która daje informajce o tym czy dany obszar zawiera element prostokątny, region poniżej oraz na prawo od aktualnego regionu.
- metody:
    - **static List<Rectangle> buildRectangles(List<Long> sizes) -** służy do tworzenia listy elementów prostokątnych z par wartości zawartych w argumencie metody. Jeśli rozmiar listy jest nieparzysty to ostatni element zostaje z niej usunięty.
    - **static void sortRectanglesDescending(List<Rectangle> rectangles)** - odpowiada za sortowanie elementów prostokątnych malejąco względem ich pól powierzchni. Wykorzystuje obiekt klasy **SortByAreaDescending**.
    - **public static List<Long> getSizesFromFile(String path)** - Odczytuje z pliku rozmiary boków elementów. Odczyt następuje linia po linii. Dane parsowane są przy pomocy wyrażenia regularnego, tak aby wyłuskać z pliku tylko niezbędne wymiary. Wymiary umieszczane są w liście i na końcu zwracane.

3.Mechanizm działania programu

Z pliku odczytywane są linia po linii wpisy zawierające dane, które są odpowiednio filtrowane przy pomocy wyrażenia regularnego, tak aby pobierane były tylko dane liczbowe. Z odczytanych par liczb tworzone są obiekty reprezentujące elementy prostokątne, które mogą zostać umieszczone na tafli. Elementy te są następnie sortowane w porządku **malejącym** (od największego do najmniejszego) względem ich pola powierzchni. Następnie tworzona jest pusta tafla o wymiarze n x n. Dla każdego prostokątnego elementu (zaczynając od tych, które mają największe pola powierzchni) odnajdywane jest odpowiednie miejsce na tafli, które pomieści dany element. Przeszukiwanie odbywa się od prawego górnego rogu afli. Początkowo tafla jest jednolita. Po umieszczeniu każdego następnego elementu, w aktualnie przeglądanej części tafli, **niewypełniona** część tafli jest dzielona na dwie części: po prawej stronie od elementu oraz poniżej elementu. W ten sposób tworzona jest struktura drzewiasta zawierająca informacje o tym gdzie znajdują się prostokątne elementy, oraz jakie pozycje są jeszcze zdatne do umieszczenia w nich prostokątów. Operacja umieszczania prostokątnych elementów na tafli jest kończona w momencie gdy algorytm przeiteruje po całej liście. Na końcu każdy prostokątny element, który został umieszczony w tafli, posiada referencję do regionu, w którym się znajduje. Następnie zliczane są pola powierzchni tych elementów, które takowe referencje posiadają. Otrzymana wartość jest odejmowana od pola powierzchni całej tafli. Poniżej znajduje się schemat dzielenia tafli i umieszczania w niej elementów prostokątnych.(Dodawane są elementy X1,X2, które dzielą obszar na A,B,C,D)


X1
A
B
X
X2
D
c
B