Laboratorium AiSD

Lista 8

Struktury drzewiaste cz. 1

Proszę pamiętać, że **część rozwiązania** zadania stanowi również **zestaw testów** zaimplementowanych algorytmów i/lub struktur danych. Dodatkowo, proszę zwracać uwagę na **powtarzające się fragmenty** kodu i wydzielać je do osobnych funkcji/klas.

1. Zdefiniuj klasy implementujące strukturę **binarnego kopca minimalnego** (*ang. binary min-heap*):
   - ***ArrayHeap<T>*** zaimplementowanego **na tablicy**,
   - ***TreeHeap<T>*** zaimplementowanego **w postaci drzewa**.

   Interfejs kopca powinien udostępniać dwie operacje:

   - *void clear()* – czyszczącą kopiec (usuwającą wszystkie elementy),
   - *void add(T element)* – wstawiającą nowy element do kopca,
   - *T minimum()* – zwracającą minimalny element kopca wraz z jego usunięciem.

   Opis budowy kopca opisano na **wykładzie 5**.

   Dla **kopca tablicowego** podczas konstrukcji przekaż **podstawową pojemność**, a gdy w kopcu nie ma miejsca, należy **rozmiar podwoić**.

   Dla **kopca drzewiastego** węzły **NIE** posiadają **informacji o rodzicu**.

   W obu przypadkach załóż, że ***null* nie jest poprawną wartością**.

2. Utwórz klasę ***PriorityQueueSorter<T>*** implementującą metodę *List<T> sort(List<T> list)* zgodnie z poniższym schematem:
   1. Wyczyść kolejkę (kopiec),
   2. Wstaw do kopca kolejne elementy z listy *list*,
   3. Zdejmuj z kopca kolejne elementy minimalne i zastępuj nimi odpowiednie elementy oryginalnej listy *list*,
   4. Zwróć listę list.

   Porównaj działanie takiego sortowania dla **obu** zaimplementowanych kopców – **mierz łączny czas** operacji z punktów 2 i 3. Pomiar czasu wykonaj wzorując się na kodzie z biblioteki dostarczonej do Listy 5.

Kopce wyświetlać na ekranie poziomami – wykorzystać kolejkę jak w metodzie **przeszukiwania wszerz** (*ang. Breadth First Search, BFS*)