Abstraktné dátové štruktúry v praxi

V tomto texte uvádzame stručný prehľad dátových štruktúr a ich dostupnosti v pár bežných jazykoch. V odhadoch časovej zložitosti predpokladáme, že operácie s prvkami (ako napr. porovnanie dvoch prvkov) vieme robiť v konštantnom čase. Premenná N všade označuje aktuálny počet prvkov uložených v dátovej štruktúre. Niektoré implementácie spĺňajú niektoré požiadavky len amortizovane (napr. vkladanie nového prvku na koniec vektora má amortizovanú časovú zložitosť). Kvôli prehľadnosti tento detail ignorujeme.

1.1 Pole s dynamickou veľkosťou (vector)

Interface:
– Indexovanie ako do poľa v $O(1)$.
– Vloženie nového prvku na koniec v $O(1)$.

C++: `vector`

Python: built-in zoznam `[]`

Java: `ArrayList` a `Vector`

1.2 Zásobník (stack)

Interface:
– Vloženie prvku v $O(1)$.
– Výber najneskôr vloženého prvku v $O(1)$.

C++: `stack`

Python: built-in zoznam `[]`

Java: `Stack`

1.3 Fronta (queue)

Interface:
– Vloženie prvku v $O(1)$.
– Výber najskôr vloženého prvku v $O(1)$.

C++: `queue`

Python: `Queue`

Java: `Queue`

1.4 Prioritná fronta (queue)

Interface:
– Vloženie prvku v $O(\log N)$.
– Výber prvku s najvyššou prioritou v $O(\log N)$.

C++: `priority_queue`

Python: `heapq`

Java: `PriorityQueue`

1.5 Obojsmerná fronta (double-ended queue)

Interface:
– Vloženie prvku na začiatok / na koniec v $O(1)$.
– Odstránenie prvku zo začiatku / z konca v $O(1)$.
– Indexovanie ako do poľa v $O(1)$.

C++: `deque`

Python: `deque`

Java: `Deque` (ale nevie indexovať)

1.6 Spájaný zoznam

Interface:
– Vloženie prvku kamkoľvek, kam máme ukazovateľ, v $O(1)$.
– Odstránenie prvku odkiaľkoľvek v $O(1)$.
– Prechod na predchádzajúci / nasledujúci prvok v $O(1)$.

C++: `list` (obojsmerný), `slist` (jednosmerný, t. j. nevieme prejsť na predch. prvok)

Python: nemá (a skoro nikdy netreba, aby mal)

Java: `LinkedList`

1.7 Usporiadaná množina (ordered set)

Interface:
– Vloženie prvku v $O(\log N)$.
– Výber prvku v $O(\log N)$.
– Test na prítomnosť prvku v $O(\log N)$.
– Výpis všetkých prvkov v utriedenom poradí v $O(N)$.

C++: `set` (a `multiset` pre situácie, kedy chceme mať násobné prvky)
Navyše máme užitočné metódy `lower_bound` a `upper_bound`, pomocou ktorých vieme efektívne nájsť všetky prvky z daného rozsahu.

Python: nemá

Java: `TreeSet`
(Máme metódy `floor` a `ceiling` analogické k tým v C++.)

1.8 Neusporiadaná množina (unordered set)

Interface:
– Vloženie prvku v $O(1)$.
– Výber prvku v $O(1)$.
– Test na prítomnosť prvku v $O(1)$.

C++: Zatiaľ nemá, bude mať v najbližšej verzii štandardu.
V `g++` už teraz nájdeme `tr1::unordered_set`.

Python: `set`

Java: `HashSet`

1.9 Asociatívne pole (map)

Interface:
– Vloženie dvojice (kľúč,hodnota) $O(1)$.
– Test na prítomnosť kľúča v $O(1)$.
– Vrátenie hodnoty priradenej kľúču v $O(1)$.

C++: `map` (usporiadané podľa kľúča, operácie sú v $O(\log N)$), `tr1::unordered_map` (neusporiadané)

Python: built-in slovník `{}`

Java: `TreeMap` (usporiadané podľa kľúča, operácie sú v $O(\log N)$), `HashMap` (neusporiadané)