Ćwiczenia 14: Drzewa przedziałowe i inne struktury

Zadanie 1. (ciemność) Dane jest n latarni ułożonych po kolei na ulicy na pozycjach od 0 do $n-1$. Proszę zaproponować strukturę danych, która zapewnia następujące operacje:

1. `turn_on(i)` – zapala i-tą latarnię,
2. `turn_off(i)` – gasi i-tą latarnię,
3. `max_darkness()` – zwraca najdłuższy przedział pozycji, na którym nie pali się żadna latarnia.

Wszystkie operacje powinny być jak najszybsze.

Zadanie 2. (następna latarnia) Sytuacja jest jak wyżej, ale zamiast operacji `max_darkness()` proszę dostarczyć operację `next(i)`, która działa następująco: jeśli i-ta latarnia jest zapalona, to zwraca następną w kolejności zapaloną żarówkę.

Zadanie 3. (wrogowie narodu) Dana jest lista życiorysów n pisarzy tworzących w Algocji. Każdy życiorys sprowadza się do przedziału $[a, b]$ gdzie a to rok rozpoczęcia działalności twórczej a b to rok jej zakończenia. Dyktator, który opanował Algocję co jakiś czas deklaruje niektórych pisarzy jako wrogów narodu a co jakiś czas ich rehabilituje. Proszę zaproponować strukturę danych z następującymi operacjami:

1. `make_enemy(i)` – i-ty pisarz zostaje wrogiem narodu,
2. `make_good(i)` – i-ty pisarz zostaje rehabilitowany,
3. `most_enemies()` – zwróć rok, w którym tworzyło najwięcej wrogów narodu.

Wszystkie operacje powinny być jak najszybsze.

Zadanie 4. (scalanie drzew czerwono-czarnych) Dane są dwa drzewa czerwono-czarne reprezentujące zbiory liczb. Proszę zaproponować algorytm obliczający drzewo czerwono-czarne będące sumą tych zbiorów.

Zadanie 5. (polujące pokémony) W n pokéballach o numerach od 0 do $n-1$ zamknięte są pokémony. Dana jest także lista węzłów typu `Hunting`, które zawierają pola `predator` i `prey`:

> Pole `predator` zawiera numer pokémona (tj. pokéballa zawierającego pokémona), który poluje na pokémona o numerze `prey`.

Jeśli pewien pokémon nie występuje na tej liście jako `predator` to znaczy, że jest spokojny. Po wypuszczeniu z pokéballa pokémon natychmiast atakuje wypuszczającego, chyba że (a) jest spokojny, lub (b) wcześniej zostały wypuszczone dwa pokémony, na które poluje. Proszę zaproponować algorytm, który sprawdza czy istnieje kolejność wypuszczania pokémonów tak, żeby żaden pokémon nie zaatakował wypuszczającego.

Zadanie 6. (wagony) Dana jest lista identyfikatorów n wagonów (każdy numer to liczba 32 bitowa). Oprócz tego mamy też listę połączeń wagonów, gdzie każde pole zawiera identyfikatory dwóch połączonych wagonów (wiadomo, że lista jest poprawna; w szczególności dany wagon może być połączony najwyżej z dwoma innymi wagonami i żaden skład nie tworzy cyklu). Proszę zaproponować algorytm obliczający długość najdłuższego ciągu wagonów.