Kopiec minmaksowy

Cała idea implementacji za pomocą jednej tablicy opiera się na prostym pomyśle: *niech parzyste elementy tablicy reprezentują jedną 'stronę' kopca, a nieparzyste drugą.* W zależności od naszych potrzeb pod indeksem 0 (pierwszym parzystym) może znajdować się odpowiednio największy element maksymalnej strony kopca bądź najmniejszy minimalnej.

1 Pomocnicze procedury, które przydadzą się później

```
zepchnij_w_dół(kopiec, idx):
    jeśli idx wskazuje na element z kopca min:
        zepchnij_w_dół_min(kopiec, idx)
    w przeciwnym wypadku:
        zepchnij_w_dół_max(kopiec, idx)

zepchnij_w_dół_min(kopiec, idx):
    syn := kopiec[2*idx + 1]
    córka := kopiec[2*idx + 2]
    wnuk1 := kopiec[2*(2*idx + 1) + 1]
    wnuk2 := kopiec[2*(2*idx + 1) + 2]
    wnuczka1 := kopiec[2*(2*idx + 2) + 1]
    wnuczka2 := kopiec[2*(2*idx + 2) + 2]
    najmniejszy := min(syn, córka, wnuk1, wnuk2, wnuczka1, wnuczka2)
    jeśli najmniejszy to syn lub córka oraz najmniejszy < kopiec[idx]:
        zamień miejscami najmniejszy oraz idx
    jeśli najmniejszy < kopiec[idx] ale nie jest synem lub córką:
        zamień miejscami najmniejszy i idx
        // aby nie zepsuć własności kopców
        jeśli rodzic najmniejszego > kopiec[idx]:
            zamień miejscami idx i rodzica najmniejszego
        zepchnij_w_dół_min(kopiec, najmniejszy)  // to dalej poziom 'min'

zepchnij_w_dół_max(kopiec, idx):
```

```
syn := kopiec[2*idx + 1]
córka := kopiec[2*idx + 2]
wnuk1 := kopiec[2*(2*idx + 1) + 1]
wnuk2 := kopiec[2*(2*idx + 1) + 2]
wnuczka1 := kopiec[2*(2*idx + 2) + 1]
wnuczka2 := kopiec[2*(2*idx + 2) + 2]
największy := min(syn, córka, wnuk1, wnuk2, wnuczka1, wnuczka2)
jeśli największy to syn lub córka oraz największy > kopiec[idx]:
    zamień miejscami największy oraz idx
jeśli największy > kopiec[idx] ale nie jest synem lub córką:
    zamień miejscami największy i idx
    jeśli rodzic największego < kopiec[idx]:
        zamień miejscami idx i rodzica największego
    zepchnij_w_dół_max(kopiec, największy)
```

Dalej bez straty ogólności przyjmuję, że zajmujemy się kopcem **min** maksowym, **maks** minowy byłby w zasadzie taki sam, z tymże w korzeniu przechowywany byłby element maksymalny.

2 Przywracanie porządku

Zakładam, że chodzi o przywrócenie porządku po dodaniu nowego elementu (na koniec).

```
popchnij_w_górę(kopiec, idx):
    jeśli idx = 0 to zakończ, wyżej nie ma już miejsca
    jeśli idx jest na poziomie minimalnym:
        // jeśli tak jest, to z pewnością nie jest najmniejszy,
        // jednak ma szansę być największy
        jeśli idx jest większy od swojego rodzica:
            zamień je miejscami
            // i jedziemy z nim maksymalnie do góry
            popchnij_w_górę_max(kopiec, rodzic)
        jeśli nie:
            popchnij_w_górę_min(kopiec, idx)
    jeśli idx jest na poziomie maksymalnym:
        // analogicznie
        jeśli idx jest mniejszy od swojego rodzica:
            zamień je miejscami
            popchnij_w_górę_min(kopiec, rodzic)
```

```
        jeśli nie:
            popchnij_w_górę_max(kopiec, idx)

popchnij_w_górę_max(kopiec, idx):
    dziadek := floor(idx-1/2)
    jeśli dziadek > 0:
        jeśli kopiec[dziadek] < kopiec[idx]
        zamień je miejscami
        popchnij_w_górę_max(kopiec, dziadek)

popchnij_w_górę_min(kopiec, idx):
    dziadek := floor(idx-1/2)
    jeśli dziadek > 0:
        jeśli kopiec[dziadek] > kopiec[idx]
        zamień je miejscami
        popchnij_w_górę_min(kopiec, dziadek)
```

3 Usuwanie minimum

```
usuń_minimum(kopiec):
    kopiec[0] := null
    zamień miejscami ostatni element kopca z dopiero co usuniętym zerowym
    zepchnij_w_dół(kopiec, 0)
```

4 Usuwanie maksimum

```
usuń_maksimum(kopiec):
    większy := 1 jeśli kopiec[1] > kopiec[2] jeśli nie to 2
    kopiec[większy] := null
    zamień miejscami ostatni element kopca z dopiero co usuniętym na indeksie większy
    zepchnij_w_dół(kopiec, większy)
```