**RB-tree = BST +**
Root, liście: B, reszta B lub R
R: wszystkie dzieci są B
**Każda ścieżka x-> liście ma tyle samo czarnych węzłów – bh(x)**
$h(T) \leq 2\lg(N+1) = O(\lg(N))$

**Insert(x)**: BST-insert(x), x.color=R, fix(x)
1) 2 R pod rząd, wujek R: recolor()
= x.parent, uncle R->B, x.grandparent = R, (recolor grandparent)
2) 2 R pod rząd, zig-zag, wujek B: rotate(x.parent) -> 3)
3) 2 R pod rząd, prosta, wujek B: rotate(x.newparent), recolor
**Delete(x)**: y=succ(x), z=y.son1, w=y.son2, BST-delete(x),fix():
1) w: R, w.child1,2: B => w.color <=> y.color, rotate(y)
2) w: B, w.child1,2: B => y<-x.color, w->RED
3) w: B, w.right: B, w.left: R => swap_col(w,w.left), rotate(w)
4) w: B, w.right: R => recolor, rotate(y)

**Deterministic select – worst O(n)**
divide A into n/k groups of k elems (k odd, k >= 5)
mediany grup: insertSort(grupa) (O(n))
M = mediana median (recursive) (T(n/5))
partition A względem M na A[1..c], A[c+1...n]
i=c: return M
i<c: select (A[1..c], i)
i>c: select (A[c+1..n], i-c) T(?)

~half of groups: at least 3 elems > M,
~half of groups: at least 3 elems < M
-> at least 3n/10 elems > M, so (?) <= 7n/10

$$T(n) \leq T\left(\frac{n}{5}\right) + t\left(\frac{7n}{10}\right) + O(n)$$

$$zał.\, T(n) \leq cn, big\ c, n > 0, O(n) \leq an$$

$$T(n) \leq c\left(\frac{n}{5}\right) + c\left(\frac{7n}{10}\right) + an$$

**Master Theorem**

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^d) = \begin{cases} \theta(n^d): d > \log_b a \\ \theta(n^d \log(n): d = \log_b a \\ \theta(n^{\log_b a}): d < \log_b a \end{cases}$$

**Radix Sort**
$\boldsymbol{n}$ liczb po $\boldsymbol{b}$ bitów - $\frac{b}{r}$ cyfr o długości $\boldsymbol{r}$ (więc tyle rund).
Zasięg: cyfry $r$ –bitowe $\rightarrow [0, 2^r - 1]$
1 runda – $O(n + 2^r)$ (count. sort)
$T_{RS}(n) = O\left(\frac{b}{r}(n + 2^r)\right) = O(n)$ – kiedy?

$f(r) = \frac{b}{r}(n + 2^r) = \frac{b}{r}n + \frac{b}{r}2^r$; zatem chcemy:

$2^r \leq n \rightarrow r = \log_2(n)$. Optymalna baza (b)?

$T_{RS} = O\left(\frac{b}{\log(n)}(n+n)\right) = O\left(\frac{bn}{\log(n)}\right)$,
przedział $\{0, 1, \dots, n^d - 1\}$

$$b = \log_n(d) \rightarrow T_{RS}(n) = O\left(\frac{\log(n^d)\, n}{\log(n)}\right) = O(dn)$$

$$\boldsymbol{d = O(1) \rightarrow T_{RS}(n) = O(n)}$$

**Random Select(A,p,r,i) – Expected O(n)**
partition A around random pivot element p into A[p...q-1], A[q,r]
k = q-p+1:
i==k: return A[q]
i<k: return RandomSelect(A,p,q-1,i)
i>k: return RandomSelect(A,q,r,i-k)

**Logarytmy**

$$log_b(mn) = \log_b(m) + \log_b(n)$$

$$\log_b\left(\frac{m}{n}\right) = \log_b(m) - \log_b(n)$$

$$\log_b(m^n) = n log_b(m)$$

$$\log_b a = \frac{\log_x a}{\log_x b}, \forall x$$

**Inwersje**
1. Merge sort: przecięcia indeksów w merge: O(nlogn)
2. Stat-tree (rb) – suma os-select(x) – idx(x): O(nlogn)

**Sumowanie pętli**

i = 1, 1+a, 1+2a, .., 1+k*a
i = 1 + ka -> k=i-1/a

j = i, i+b, i+2b, ... 1+m*a
j = 1+ma -> m = j-1/a

```
for i = 1..n
  j=i
  for j = i..n
    P(i,j)
    j += b
  i += a
```

**Longest Common Subsequence - O(mn)** $n$
X[1..n],Y[1..m]
c[i,j] = |LCS(X[1,,i],Y[1,..,j])| -> tablica [n+1][m+1] (0,_)=(_,0)=0
odczyt od c[n,m], idziemy po równych wartościach ([-1,0],[0,-1]), jak nie ma -> ta komórka jest w LCS, skaczemy [-1,-1]
**DUŻO ANALOGICZNYCH**

**Edit distance O(mn)**
X[1,...,n], Y[1,...,m]
$\mathrm{e[i,j]} = \min\{1 + \mathrm{E(i-1,j)}, 1 + \mathrm{E(i,j-1)}, \mathrm{diff(i,j)} + \mathrm{E(i-1,j-1)}\}$
->tablica[m][n]
diff(x,y) =1 ⇔ x(i) != y(i), else 0
odczyt jak LCS

**Wycieczka**

Po drodze mamy $n$ hoteli, $i$ -ty hotel w odległości $a_i$ od startu. Wiemy, że $a_1 < a_2 < \ldots < a_n$. Po drodze możemy zatrzymywać się tylko w tych podanych hotelach, ale możemy wybrać w których. $n$ -ty hotel jest w miejscu docelowym, tam musimy przenocować. Chcemy przebyć do 600km dziennie (ale możemy mniej, nie więcej). Pokonanie $x$ km danego dnia kosztuje nas $\frac{(600-x)^{13}}{x}$ za ten dzień. Chcemy zaplanować podróż tak, aby zminimalizować sumę opłat za wszystkie dni. Podaj algorytm (o złożoności co najwyżej wielomianowej), który wyznaczy optymalny ciąg hoteli w których należy się zatrzymać. Udowodnij poprawność i złożoność obliczeniową.

$$C(n) = min\left\{C(v) + \frac{(600 - d(v,n))^{13}}{d(v,n)}\right\}$$
$$v : d(v,n) \leq 600$$
$$v < n$$

**Stacje benzynowe**

Jedziemy przez paliwową pustynię pojazdem palącym 1 litr paliwa na 1km. Pojemność baku wynosi $W$. Znamy rozkład stacji benzynowych wzdłuż drogi, którą jedziemy i wszystkie one są położone na pełnych kilometrach. Na stacji numer $i$, cena paliwa wynosi $w_i$. Pokaż algorytm obliczający jak najtaniej można dojechać do końca drogi (czyli stacji n).

$L(k,t)$ – minimalny koszt dojazdu do $k$ -tej stacji z $t$ litrami

$$L(i,t)=\min_{\substack{j<k \\ |k_i-k_j|\leq w}} \{L(j,x)+w_j(k_i-k_j-x+t$$

**Warunki drzewa czerwono-czarnego**

Drzewo BST jest drzewem czerwono-czarnym, jeśli ma następujące własności:

1) Każdy węzeł jest albo czerwony, albo czarny.
2) Każdy liść bezwartościowy *nil* jest czarny.
3) Jeśli węzeł jest czerwony, to oba jego synowie są czarni.
4) Każda prosta ścieżka z ustalonego węzła do liścia ma tyle samo czarnych węzłów (czarna wysokość).

**Lemat**

Drzewo RB o $n$ węzłach wewnętrznych ma wysokość co najwyżej $2lg(n+1)$.

**Najdłuższy wspólny podciąg**

Zdefiniujmy $c[i,j]$ jako długość NWP ciągów $X_i$ i $Y_i$.
Jeśli $i=0$ lub $j=0$, to jeden z ciągów ma długość $0$, ich NWP ma długość $0$.
Z optymalnej struktury wynika następująca zależność rekurencyjna:

$$c[i,j]=\begin{cases} 0 \; jeśli \, i=0 \, lub \, j=0 \\ c[i-1,j-1] \; jeśli \, i,j>0 \, i \, x_i=y_i \\ max(c[i,j-1],c[i-1),j] \; jeśli \, i,j>0 \, i \, x_i \neq y_i \end{cases}$$

| **ALGORYTM** | **BEST** | **AVERAGE** | **WORST** | **WORST SPACE** |
|---|---|---|---|---|
| Insertion Sort | $O(n)$ | $O(n^2)$ | $O(n^2)$ | |
| Merge Sort | $O(nlogn)$ | $O(nlogn)$ | $O(nlogn)$ | $O(n)$ |
| Quick Sort | $O(nlogn)$ | $O(nlogn)$ | $O(nlogn)$ | $O(n)$ |
| Heap Sort | $O(nlogn)$ | $O(nlogn)$ | $O(nlogn)$ | Heap |
| Counting Sort | $O(n+k)$ | $O(n+k)$ | $O(n+k)$ | *k - przedział |
| Radix Sort | $O(d(n+k$ | d – ilość cyfr | n – ilość liczb | k – i różnych cyfr |
| | | | | |
| Heapify | $O(lgn)$ | $T(n)\leq T(2n/3)+\Theta(1)=$ | | |
| Build Heap | $O(n)$ | | | |
| Heap Insert | $O(lgn)$ | | | |
| Extract Max | $O(lgn)$ | | | |
| | | | | |
| Quick Select | $O(n)$ | $O(n)$ | $O(n^2)$ | |
| Mediana median | $O(n)$ | $O(n)$ | $O(n)$ | |
| | | | | |
| BST Sort | $O(nlogn)$ | $O(nlogn)$ | $O(n^2)$ | $\Theta(n)$ |
| Search, Minimum, Maximum, Successor, Predecessor Insert oraz Delete | | | $O(h)$ | h - wysokość |
| OS-Select | $O(h)=O($ | | | |
| OS-Rank | $O(h)=O($ | | | |