(c) drzewa RST,
(d) drzewa TRIE,
(e) drzewa PATRICIA.

Narysuj odpowiednie drzewa.

3.24. Zaprojektuj strukturę danych umożliwiającą wykonywanie w czasie $O(\log n)$ następujących operacji na zbiorze S:

(a) *makeset*(S):: $S := \varnothing$;
(b) *insert*((x, y), S):: $S := S \cup \{(x, y)\}$;
(c) *minx*(S):: usunięcie z S pary (x, y) o najmniejszej pierwszej składowej;
(d) *miny*(S):: usunięcie z S pary (x, y) o najmniejszej drugiej składowej;
(e) *searchx*(x, S):: wyznaczenie takiej pary (a, b), że $x = a$;
(f) *searchy*(y, S):: wyznaczenie takiej pary (a, b), że $y = b$.

Elementy składowe par z S pochodzą ze zbiorów liniowo uporządkowanych. Wyznacz złożoność pamięciową implementacji.

3.25. Zaprojektuj strukturę danych umożliwiającą wykonywanie w czasie $O(\log n)$ następujących operacji na zbiorze S:

(a) *construct*(S):: utworzenie ciągu pustego S;
(b) *insert*(S, x):: $S := S \cup \{x\}$;
(c) *delete*(S, x):: $S := S - \{x\}$;
(d) *search*(S, x}:: sprawdzenie, czy x znajduje się w zbiorze S;
(e) *elem*(S, i):: wyznaczenie i-tego co do wielkości elementu zbioru S;
(f) *numb*(S, x):: wyznaczenie numeru elementu x w zbiorze S (względem wielkości).

Zakładamy, że elementy zbioru S pochodzą z dowolnego liniowo uporządkowanego uniwersum U.

3.26. Zaproponuj efektywną strukturę danych do wykonywania ciągów następujących operacji (dla elementów x pochodzących z dowolnego zbioru liniowo uporządkowanego):

(a) *initialization*:: $S_i := \varnothing$ dla $i = 1, 2, \ldots, n$;
(b) *insert*(x, S_i):: $S_i := S_i \cup \{x\}$ pod warunkiem, że x nie występuje w żadnym zbiorze S_j, $1 \le j \le n$;
(c) *deletemin*(S_i):: usunięcie ze zbioru S_i najmniejszego elementu;
(d) *find*(x):: wyznaczenie numeru zbioru, do którego należy element x.

Jaka jest złożoność operacji?

3.27. Zaprojektuj strukturę danych umożliwiającą wykonywanie w czasie $O(\log n)$ następujących operacji na ciągu S:

(a) *construct*(S):: utworzenie ciągu pustego S;
(b) *insert*(S, i, x):: wstawienie x na i-te miejsce w ciągu S, tzn. $S_i := x$, pod warunkiem, że $i \le |S| + 1$;
(c) *sum*(S, i, j):: obliczenie sumy $\sum_{k=i}^{j} S_k$.

Zakładamy, że elementy ciągu są liczbami całkowitymi.

3.28. Do operacji słownika *search*, *insert* i *delete* dodajemy operację:

$$between(x, y) = |\{a \in S: x \le a \le y\}|$$

Podaj implementację słownika, przy której każda operacja ma pesymistyczną złożoność czasową $O(\log n)$.

3.29. Zaprojektuj strukturę danych umożliwiającą wykonywanie w czasie $O(\log n)$ następujących operacji na zbiorze S zawierającym przedziały liczb rzeczywistych $[l, r]$:

(a) *empty*(S):: $S := \varnothing$;
(b) *add*(S, I):: $S := S \cup \{I\}$;
(c) *delete*(S, I):: $S := S - \{I\}$;
(d) *is*(S, x):: sprawdzenie, czy element x należy do jakiegoś przedziału zbioru S;
(e) *intersect*(S, I):: sprawdzenie, czy przedział I ma niepuste przecięcie z jakimś przedziałem należącym do S.