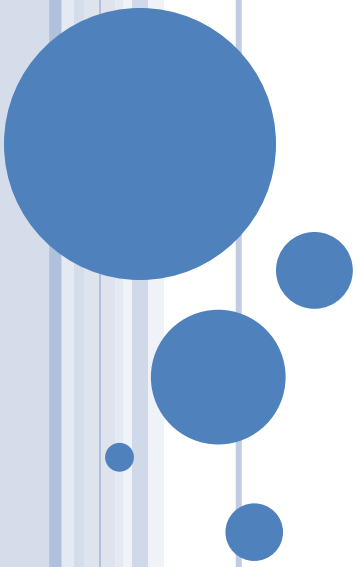


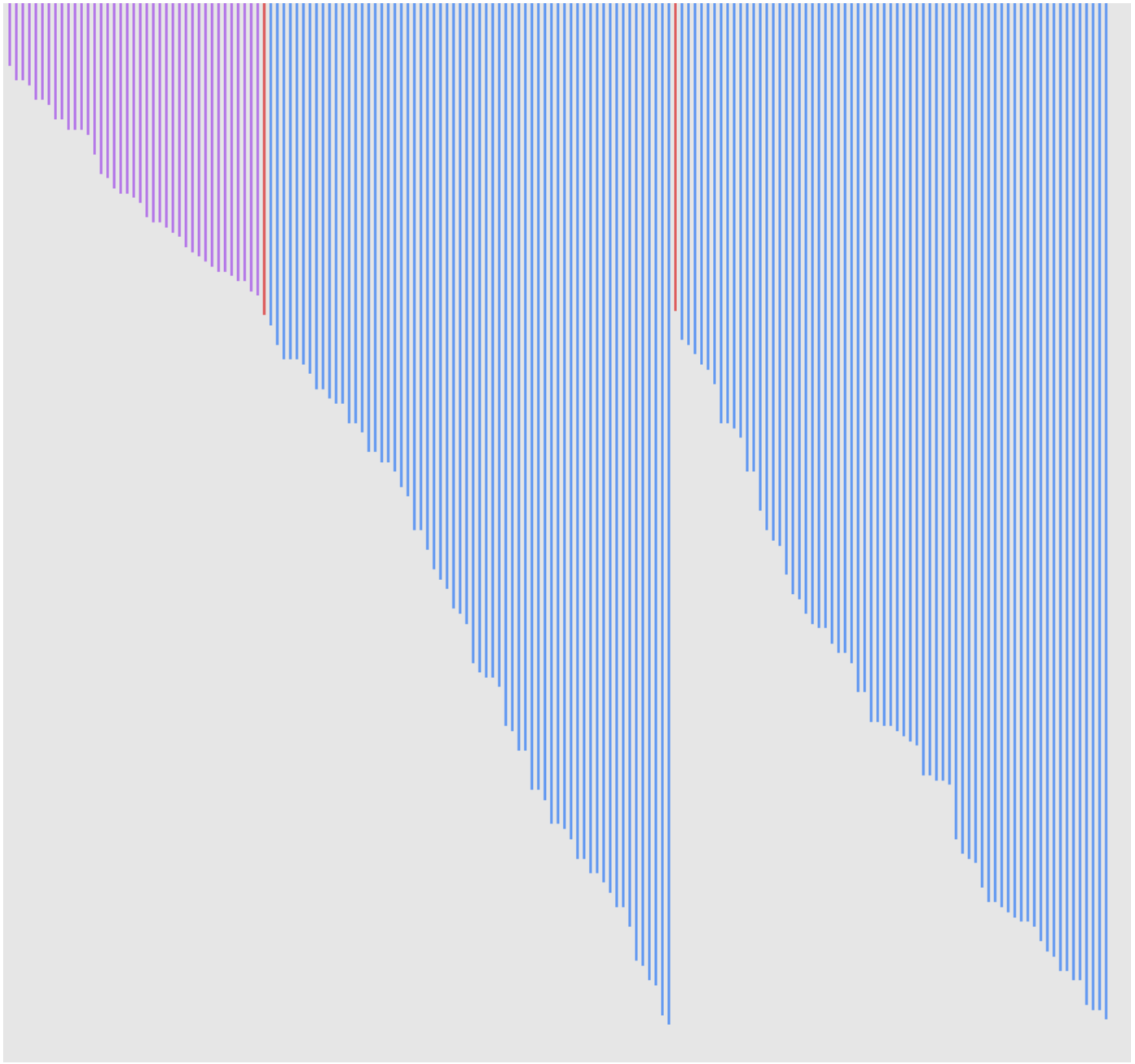
Triedenia

Mergesort, výberom, haldové

Výpočtová zložitosť problému

Triedenie zlučovaním (mergesort)

- metóda **„rozdeľuj a panuj“**
  - postupnosť sa rozdelí na dve polovice, ktoré sa utriedia (rozdeľuj)
  - utriedené polovice sa zlúčia do jednej utriedenej postupnosti (panuj)
- vhodné aj na triedenie súborov
- varianty: dvojcestné, viaccestné zlučovanie



TRIEDENIE ZLUČOVANÍM

```
void mergeSort(int a[], int l, int r)
{
  if  (l >= r) return;

  int s = (l+r)/2;

  mergeSort(l,s);
  mergeSort(s+1,r);

  //zlučovanie
  i = l;  j = s+1;  k = l;
  while ((i ≤ s) and (j ≤ r)) {
      if (a[i] < a[j]) {
        b[k] = a[i]; i++;
      } else {
        b[k] = a[j]; j++;
      }
      k++;
  }
  while (i ≤ s) {
     b[k] = a[i]; i++; k++;
  }
  while (j ≤ r) {
     b[k] = a[j]; j++; k++;
  }
  for (i = l; i ≤ r ; i++) {
     a[i] = b[i];
  }
}
```

Príklad výpočtu

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
3	**1**	**4**	**5**	**9**	**10**	**11**	**6**	**2**	**14**	**15**	**7**	**8**	**12**	**13**
3	1	4	5	9	10	11	6	2	14	15	7	8	12	13
3	1	4	5	9	10	11	6	2	14	15	7	8	12	13
3	1	4	5	9	10	11	6	2	14	15	7	8	12	13
3	1	4	5	9	10	11	2	6	14	15	7	8	12	13
1	3	4	5	9	10	11	2	6	14	15	7	8	12	13
1	3	4	5	9	10	11	2	6	7	8	12	13	14	15
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

- počet porovnaní je 35 — $O(n \log_2 n)$
- počet presunov je 118 — $O(2\, n \log_2 n)$

Najhorší prípad

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
8	**4**	**12**	**2**	**10**	**6**	**14**	**1**	**9**	**5**	**13**	**3**	**11**	**7**	**15**
8	4	12	2	10	6	14	1	9	5	13	3	11	7	15
8	4	12	2	10	6	14	1	9	5	13	3	11	7	15
8	4	12	2	10	6	14	1	9	5	13	3	11	7	15
8	4	12	2	10	6	14	1	9	5	13	3	11	7	15
4	8	12	2	6	10	14	1	5	9	13	3	7	11	15
2	4	6	8	10	12	14	1	3	5	7	9	11	13	15
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

- počet porovnaní je 45
- počet presunov je 118

Výpočtová zložitosť algoritmu Mergesort

- pamäťová
  - $S(n) = O(2n+\log n)$
  - na zlučovanie je potrebná pomocná pamäť veľkosti vstupnej postupnosti – spolu $O(2n)$ a zásobník veľkosti $O(\log n)$
- časová
  - $T(n) = O(n \log n)$

Triedenie výberom

- Výberom maxima,
- Výberom minima,
  - $T(n)=O(n^2)$
- Haldové triedenie – čiastočné usporiadanie poľa do haldy, v ktorej je maximum v koreni
  - $T(n)=O(n.\log(n))$

Triedenie výberom maxima

```
int imax, p;
for (int j = n - 1; j >= 1; j--) {
   imax = j;
   for (int i = 0; i <= j - 1; i++) {
      if (a[i] > a[imax]) {
         imax = i;
      }
   }
   p = a[imax];
   a[imax] = a[j];
   a[j] = p;
}
```

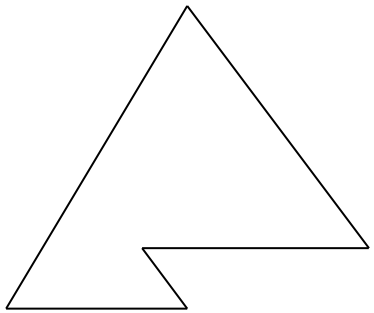
Triedenie výberom minima

```
int imin, p;
for (int j = 0; j < n - 1; j++) {
   imin = j;
   for (int i = n - 1; i > j; i--) {
      if (a[i] < a[imin]) {
         imin = i;
      }
   }
   p = a[imin];
   a[imin] = a[j];
   a[j] = p;
}
```

Haldové triedenie

Halda je binárny strom s vlastnosťami:

- Usporiadanie – rodič ≥ potomok
- Tvar – listy sú najviac v dvoch úrovniach čo najviac vľavo



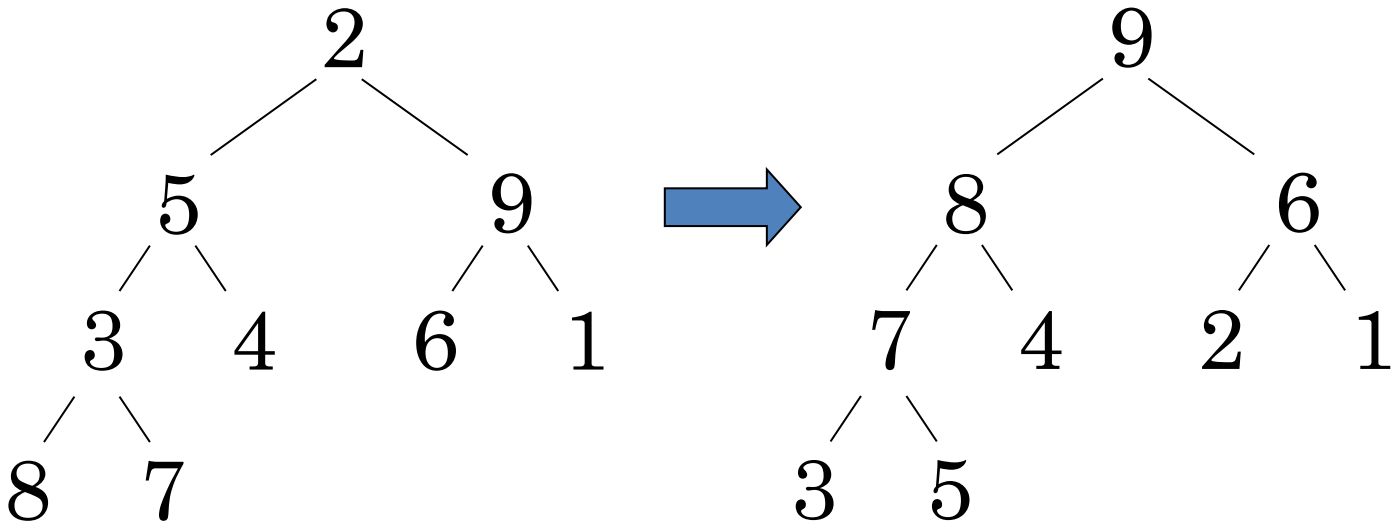
Implementácia haldy v poli

Nech $a[1..n]$

- $a[i]$ je rodič $a[2i]$, $a[2i+1]$,
- $a[i] \geq a[2i]$, $a[i] \geq a[2i+1]$,
- $a[i]$ je potomok $a[i/2]$,
- $a[1]$ je koreň a maximum.

1 2 3 4 5 6 7 8 9

a=(2, 5, 9, 3, 4, 6, 1, 8, 7)

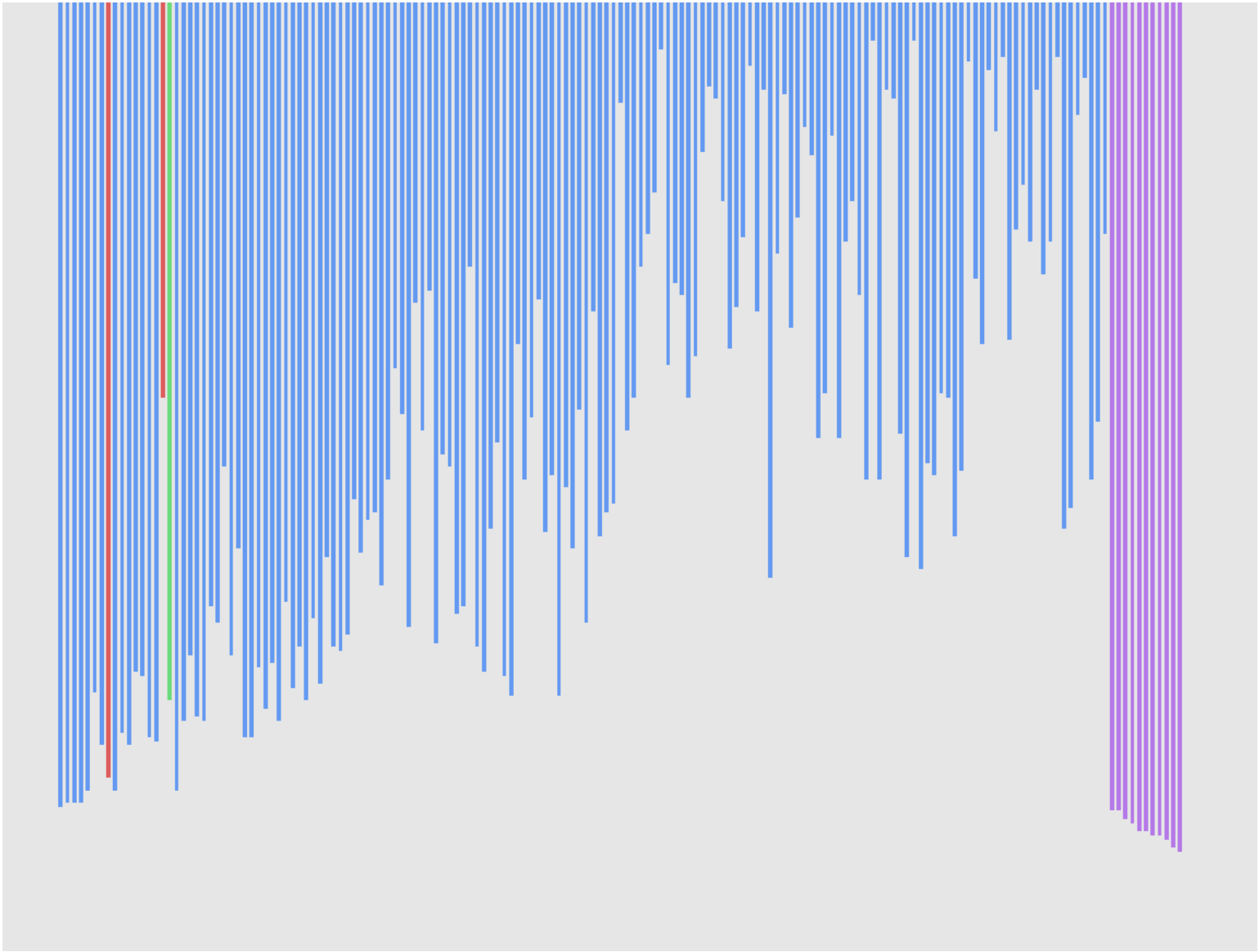


1 2 3 4 5 6 7 8 9

a=(9, 8, 6, 7, 4, 2, 1, 3, 5)

Haldové triedenie

- Vytvorenie haldy O(n.log n),
- Opakuj O(n) krát:
  - výmena maxima z koreňa s prvkom na konci haldy, zníženie počtu prvkov haldy, O(1)
  - obnovenie haldy O(log n)



VYTVORENIE HALDY

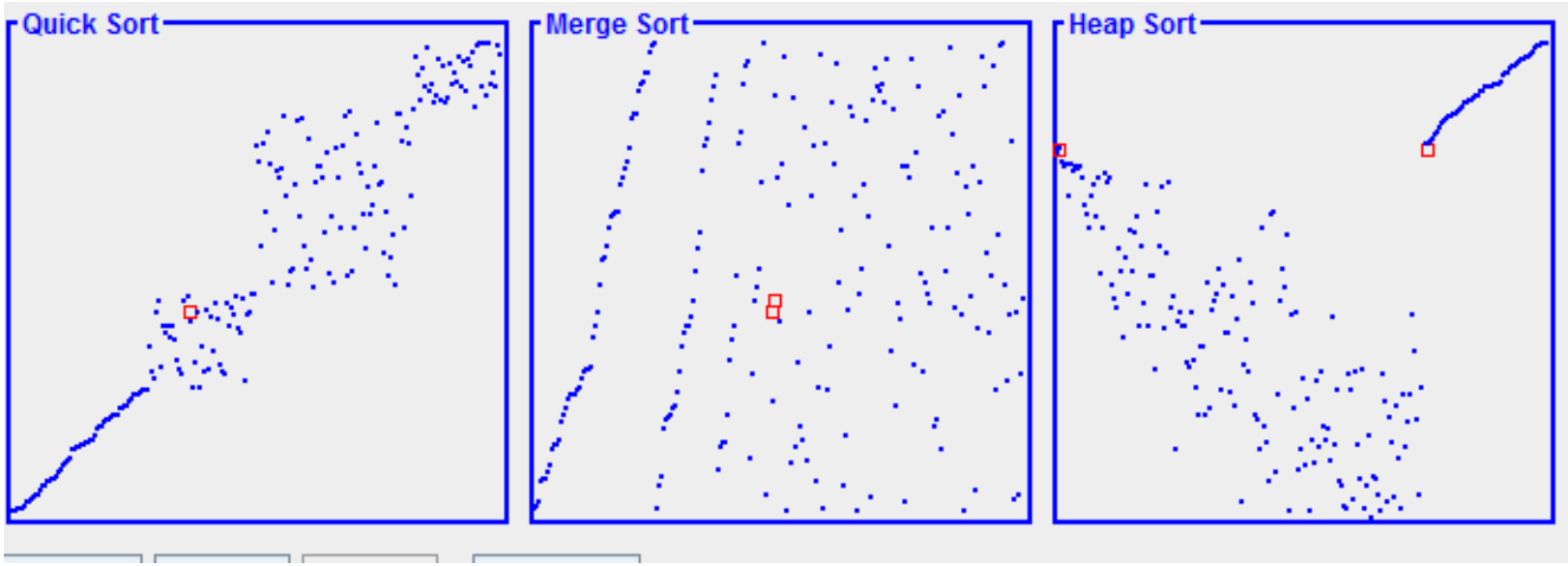
```
void heapify(int l, int r) {

  int p, ch;
  p = l; ch = 2*p;
  if ch + 1 ≤ r {
    if a[ch + 1] > a[ch] {ch++};
  }
  while ch ≤ r {
    if a[p] < a[ch] {
        výmena (p,ch);
        p = ch;
        ch = 2*p;
        if ch+1 ≤ r {
           if a[ch+1] > a[ch] {ch++;}
        }
    } else {ch = r + 1;}
  }
}
```

Triedenie haldou

```
void heapsort;

for (int i = n / 2; i > 0; i++) {
   heapify(i,n);
}
for (int i = n; i > 1; i-- ){
   výmena(a[1], a[i]);
   heapify(1,i-1)
}
```


Quick Sort
Merge Sort
Heap Sort

Výpočtová zložitosť problému

Časová výpočtová zložitosť $T_P(n)$ (alebo pamäťová $S_P(n)$) problému P rozsahu n je minimum z výpočtových zložitostí $T_A(n)$ (alebo $S_A(n)$) všetkých algoritmov A, ktoré riešia problém P.

- Napríklad: P je problém triedenia, jeho časová výpočtová zložitosť je minimum zo zložitostí všetkých triediacich algoritmov, napr. bubblesort $O(n^2)$, selectsort $O(n^2)$, mergesort $O(n.\log n)$, quicksort $O(n.\log n)$, heapsort $O(n.\log n)$.
- Časová výpočtová zložitosť problému triedenia je $O(n \log n)$.
- Časová výpočtová zložitosť problému triedenia je $\Omega(n \log n)$.