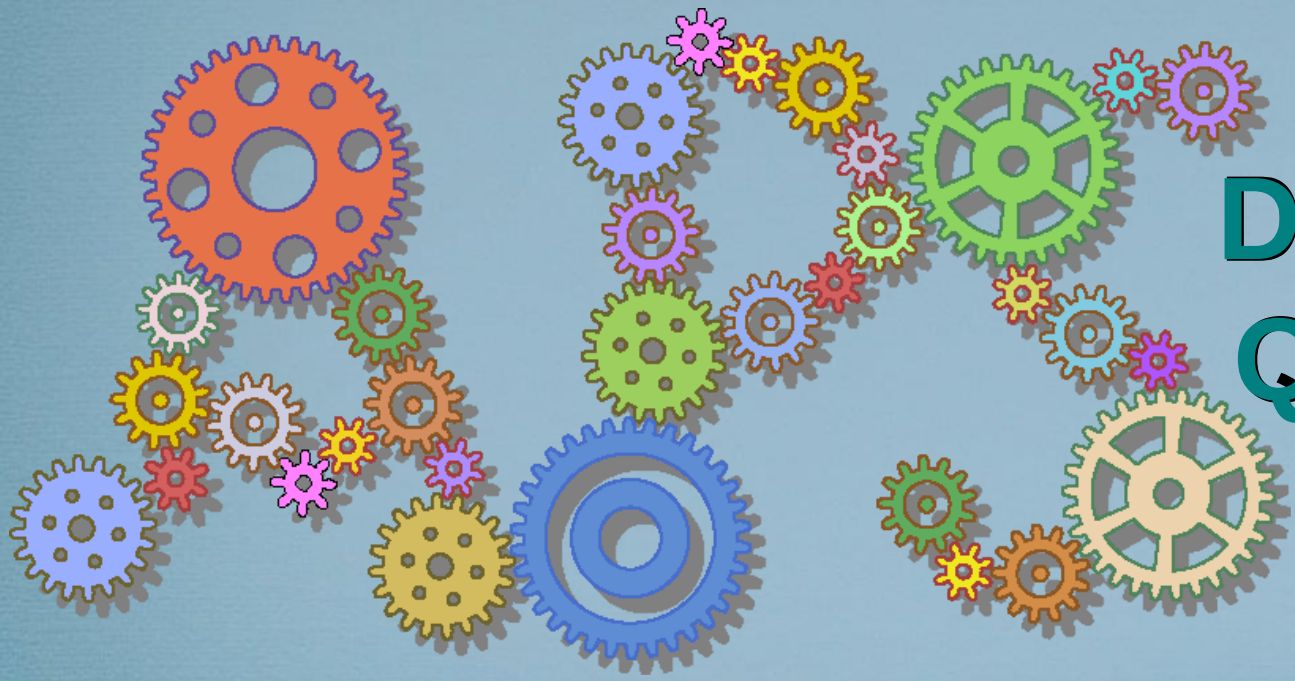
Algoritmi in podatkovne strukture



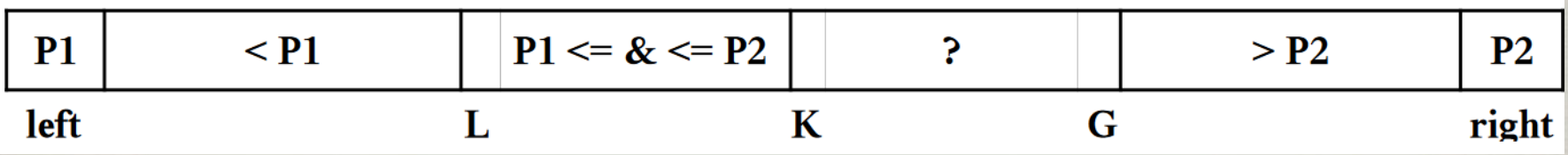
Dual-Pivot Quicksort

Navaden Quicksort

- Na seznamih primitivnih tipov
- Izberemo pivot
- Gremo čez seznam in naredimo dve particiji
  - Elementi manjši od pivota
  - Elementi večji od pivota
- Rekurzivno ponovimo na vsaki particiji

Dva pivota

- Z uporabo dveh pivotov razdelimo seznam na tri particije
- Potrebujemo tri oznake: L, K, G



Postopek

- Izberemo dva pivota (recimo prvi in zadnji element seznama). P1 naj bo manjši od P2, sicer ju zamenjamo.
- V prvem delu seznama bodo elementi manjši od P1
- V drugem delu bodo elementi večji od P1 in manjši od P2
- Sledijo elementi, ki jih še moramo obdelati
- V zadnjem delu bodo elementi, ki so večji od P2

Postopek

- Vzamemo prvi element v neobdalenem delu, ga primerjamo s pivotoma in damo v primeren del
- Skladno spremenimo oznake L, K, G
- Zgornja koraka ponavljamo dokler K≤G
- P1 zamenjamo z zadnjim elementom v prvem delu
- P2 zamenjamo s prvim elementom v zadnjem delu
- Rekurzivno ponovimo na vseh treh particijah

Število operacij

- Navadni Quicksort:
  - Število primerjav: 2*n*ln(n)
  - Število zamenjav: 1*n*ln(n)
- Dual-Pivot Quicksort:
  - Število primerjav: 2*n*ln(n)
  - Število zamenjav: 0,8*n*ln(n)

Psevdokoda

```
DUALPIVOTQUICKSORT(A, left, right)  // sort A[left..right]
 1  if right − left ≥ 1
        // Take outermost elements as pivots (replace by sampling)
 2      p := min{A[left], A[right]}
 3      q := max{A[left], A[right]}
 4      ℓ := left + 1;  g := right − 1;  k := ℓ
 5      while k ≤ g
 6          if A[k] < p
 7              Swap A[k] and A[ℓ];  ℓ := ℓ + 1
 8          else if A[k] ≥ q
 9              while A[g] > q and k < g
10                  g := g − 1
11              end while
12              Swap A[k] and A[g];  g := g − 1
13              if A[k] < p
14                  Swap A[k] and A[ℓ];  ℓ := ℓ + 1
15              end if
16          end if
17          k := k + 1
18      end while
19      ℓ := ℓ − 1;  g := g + 1
20      A[left] := A[ℓ];  A[ℓ] := p     // p to final position
21      A[right] := A[g];  A[g] := q  // q to final position
22      DUALPIVOTQUICKSORT(A, left , ℓ − 1)
23      DUALPIVOTQUICKSORT(A, ℓ + 1, g − 1)
24      DUALPIVOTQUICKSORT(A, g + 1, right)
25  end if
```

Lastnosti

- Princip deli in vladaj
- In-place sortiranje
- Hitrejši od QS