NIZOVI I SORTIRANJE

Counting Sort

- Pretpostavke:
  - Elementi niza su celi brojevi
  - Vrednosti elemenata niza se nalaze u opsegu [0,k]
- Složenost algoritma je linearna O(n)
- Zahteva dodatni memorijski prostor veličine n+k
- Ideja:
  - Ako znamo broj elemenata niza (m) koji su manji od nekog elementa x, onda x treba smestiti na poziciju m

Counting Sort

```
void CountingSort(int a[], int b[], int n, int k) {
   int *c = new int[k];
   int i;
   for (i=0; i<k; i++)
       c[i] = 0;
   for (i=0; i<n; i++)
       c[a[i]]++;
   c[0]--;
   for (i=0; i<k-1; i++)
       c[i+1] += c[i];
   for (i=n-1; i>=0; i--) {
       b[c[a[i]]] = a[i];
       c[a[i]]--;
   }
   delete[] c;
}
```

Counting Sort

- Kada treba koristiti ovaj algoritam za sortiranje?
- Kada je opseg vrednosti elemenata niza znatno manji ili približno jednak od veličine niza

Izdvajanje i-tog elementa niza

- Izdvojiti i-ti po veličini element neuređenog niza
- Algoritam: Randomized Select
- Koristi podelu niza na dva dela kao kod QuickSort-a
- Prosečna složenost O(n)

Izdvajanje i-tog elementa niza

```
int RandomizedPartition(int data[], int first, int last) {
   int ind = first + (last-first) * rand() / (double) RAND_MAX;
   Swap(data, first, ind);
   int lower = first+1, upper = last;
   int bound = data[first];
   while( lower <= upper ){
        while( data[lower] < bound && lower < last )  lower++;
        while( bound < data[upper] && upper > first ) upper--;
        if( lower < upper )
                Swap(data, lower++, upper--);
        else
                lower++;
   }
   Swap(data, first, upper);
   return upper;
}
```

Izdvajanje i-tog elementa niza

```
int RandomizeSelect(int data[], int first, int last, int i)
{
   if (first == last)
        return data[first];
   int part = RandomizedPartition(data, first, last);
   int k = part - first + 1;
   if (i == k)
        return data[part];
   else if (i < k)
        return RandomizeSelect(data, first, part-1, i);
   else
        return RandomizeSelect(data, part+1, last, i-k);
}
```

Rešavanje sistema linearnih jednačina

- Sistem jednačina: $A \cdot x = b$
- Metod: **LUP dekompozicija**
- Preduslov: Matrica A nema singularitet
- Zadatak: Odrediti matrice L, U i P tako da važi $P \cdot A = L \cdot U$
- L je donja trougaona matrica, U je gornja trougaona matrica, P je permutaciona matrica
- $P \cdot A \cdot x = P \cdot b \Rightarrow L \cdot U \cdot x = P \cdot b$
- $L \cdot y = P \cdot b$
- $U \cdot x = y$

Rešavanje sistema linearnih jednačina

- Sistem linearnih jednačina se svodi na rešavanje dva sistema linearnih jednačina
  - Prvog donje trougaonog sistema (foreward substitution)

$$y_i = b_{\pi[i]} - \sum_{j=1}^{i-1} l_{ij} y_j$$

  - Drugog gornje trougaonog sistema (backward substitution)

$$x_i = \left(y_i - \sum_{j=i+1}^{n} u_{ij} x_j \right) / u_{ii}$$

Rešavanje sistema linearnih jednačina

```
void LUPDecomposition(double** a,
                      int n, int* p) {
    int i, j, k, ind, itmp;
    double max, tmp;
    for (i=0; i<n; i++)
        p[i] = i;
    for (k=0; k<n; k++) {
        max = 0;
        for (i=k; i<n; i++)
            if (max < fabs(a[i][k]))
   {
                max = fabs(a[i][k]);
                ind = i;
            }
        if (max == 0)
            throw "Singulatitet";
        if (ind != k) {
            itmp = p[k];
            p[k] = p[ind];
            p[ind] = itmp;
            for (i=0; i<n; i++) {
                tmp = a[k][i];
                a[k][i] = a[ind][i];
                a[ind][i] = tmp;
            }
        }
        for (i=k+1; i<n; i++) {
            a[i][k] /= a[k][k];
            for (j=k+1; j<n; j++)
                a[i][j] -= a[i][k] *
                           a[k][j];
        }
    }
}
```

Rešavanje sistema linearnih jednačina

```
void LUPSolver(double** a, int n, double* b, double* x) {
   int i, j;
   int* p = new int[n];
   LUPDecomposition(a, n, p);
   double* y = new double[n];
   for (i=0; i<n; i++) {
        y[i] = b[p[i]];
        for (j=0; j<i; j++) {
                y[i] -= a[i][j] * y[j];
        }
   }
   for (i=n-1; i>=0; i--) {
        x[i] = y[i];
        for (j=i+1; j<n; j++) {
                x[i] -= a[i][j] * x[j];
        }
        x[i] /= a[i][i];
   }
   delete[] y;
   delete[] p;
}
```