Universiteti i Prishtinës

Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike

Algoritmet dhe struktura e të dhënave

Vehbi Neziri

FIEK, Prishtinë 2015

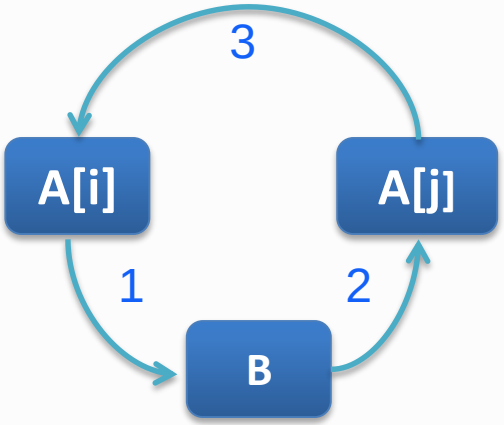
- Algoritmet e sortimit
  - Buble
  - Selection
  - Insertion
  - Quick
  - Merge
  - Radix
  - Shell

Sortimi Bubble

- Krahaso secilin element (përveç të fundit) me fqinjët në të djathtë
  - Nëse nuk janë të sortuar, shkëmbe
  - Vendos elementin më të madh në fund
  - Elementi i fundit tashmë është në vendin e duhur
- Krahaso secilin element (përveç dy të fundit) me fqinjët në të djathtë
  - Nëse nuk janë të sortuar, shkëmbe
  - Vendos elementin më të madh pas të fundit
  - Dy elementet e fundit tashmë është në vendin e duhur
- Krahaso secilin element (përveç tre të fundit) me fqinjët në të djathtë
  - Vazhdo si më sipër deri sa të kryhet sortimi

Shembull i sortimit Bubble

A	5	1	**12**	-5	16	→	Vektori i pasortuar
1	5	1	**12**	-5	16	→	5>1, ndërro
	1	5	12	-5	16	→	5>12, jo
	1	5	12	-5	16	→	12>-5, ndërro
	1	5	-5	12	16	→	12>16, jo
2	1	5	**-5**	12	16	→	1>5, jo
	1	5	-5	12	16	→	5>-5, ndërro
	1	-5	5	12	16	→	5>12, jo
3	1	-5	**5**	12	16	→	1>-5, ndërro
	-5	1	5	12	16	→	1>5, jo
4	-5	1	5	12	16	→	-5>1, jo
B	-5	1	5	12	16	→	Vektori i sortuar



Shembull 13.1

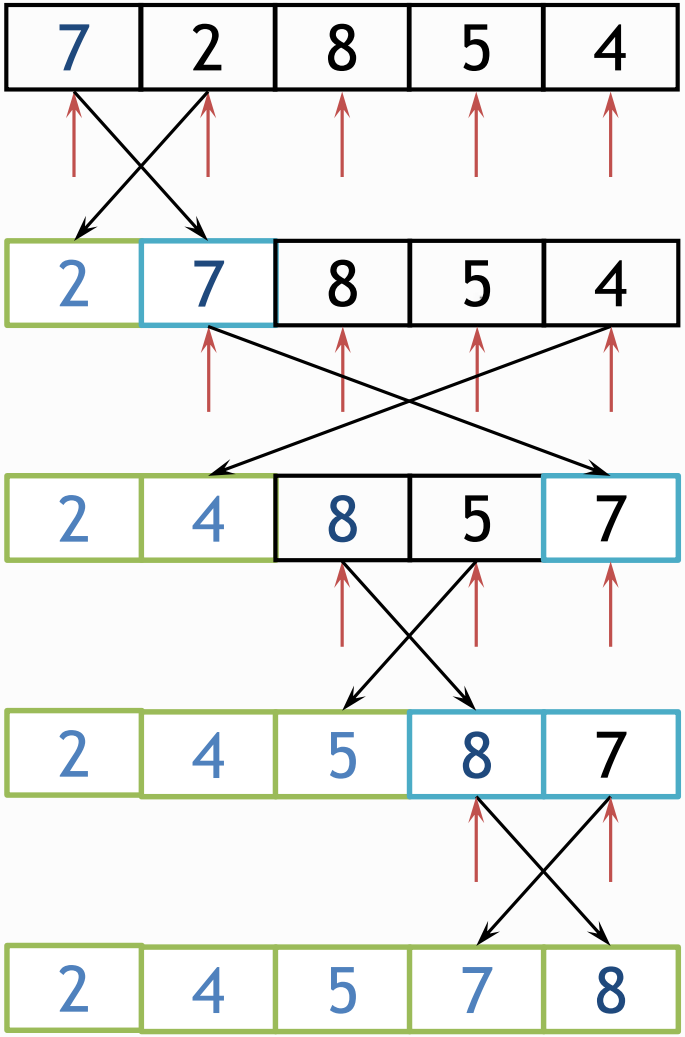
- Të shkruhet programi për sortimin e një vargu duke përdorur algoritmin e sortimit Bubble.

```
1  #include<iostream>
2  using namespace std;
3  void bubbleSort(int vargu[], int n)
4  {
5      int perk;
6      for (int i = 0; i<4; i++)
7      {
8          for (int j = 0; j<4; j++)
9          {
10             if (vargu[j]>vargu[j + 1])
11             {
12                 perk = vargu[j];
13                 vargu[j] = vargu[j + 1];
14                 vargu[j + 1] = perk;
15             }
16         }
17     }
18 }
19 void shtyp(int vargu[], int m)
20 {
21     for (int i = 0; i < m; i++)
22         cout << vargu[i] << " ";
23     cout << "\n";
24 }
25 int main()
26 {
27     int vargu[] = { 5, 1, 12, -5, 16 };
28
29     cout << "Vargu origjinal i pasortuar:\n";
30     shtyp(vargu, 5);
31
32     bubbleSort(vargu, 5);
33
34     cout << "Vargu i sortuar: " << endl;
35     shtyp(vargu, 5);
36
37     system("pause");
38     return 0;
39 }
```

Sortimi selection

- Për një varg të dhënë
  - Kërko elementet nga 0 deri në n-1 dhe përzgjidh më të voglin.
    - Shkëmbe me elementin në lokacionin 0.
  - Kërko elementet nga 1 deri në n-1 dhe përzgjidh më të voglin.
    - Shkëmbe me elementin në lokacionin 1.
  - Kërko elementet nga 2 deri në n-1 dhe përzgjidh më të voglin.
    - Shkëmbe me elementin në lokacionin 2.
  - Përseritet veprimi deri sa të arrihet në fund.

Sortimi selection



Shkëmbe 7 me 2

Shkëmbe 7 me 4

Shkëmbe 8 me 5

Shkëmbe 8 me 7

Shembull 13.2

- Të shkruhet programi për sortimin e një vargu duke përdorur algoritmin e sortimit Selection.

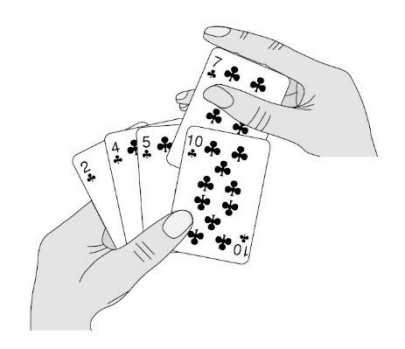
```
1  #include<iostream>
2  using namespace std;
3  void selectionSort(int vargu[], int m)
4  {
5      int i, j, min,perk;
6
7      for (i = 0; i < m - 1; i++)
8      {
9          min = i;
10         for (j = i + 1; j < m; j++)
11             if (vargu[j] < vargu[min])
12                 min = j;
13
14         perk = vargu[min];
15         vargu[min] = vargu[i];
16         vargu[i] = perk;
17     }
18 }
19 void shtyp(int vargu[], int m)
20 {
21     for (int i = 0; i < m; i++)
22         cout << vargu[i] << " ";
23     cout << "\n";
24 }
25 int main()
26 {
27     int vargu[] = { 5, 1, 12, -5, 16 };
28     int n = sizeof(vargu) / sizeof(vargu[0]);
29     cout << "Vargu origjinal: \n";
30     shtyp(vargu, n);
31
32     selectionSort(vargu, n);
33
34     cout << "\nVargu i sortuar: \n";
35     shtyp(vargu, n);
36     system("Pause");
37     return 0;
38 }
```

Sortimi Insertion

- Sortimi Insertion i takon grupit të algoritmeve të sortimit O(n2).

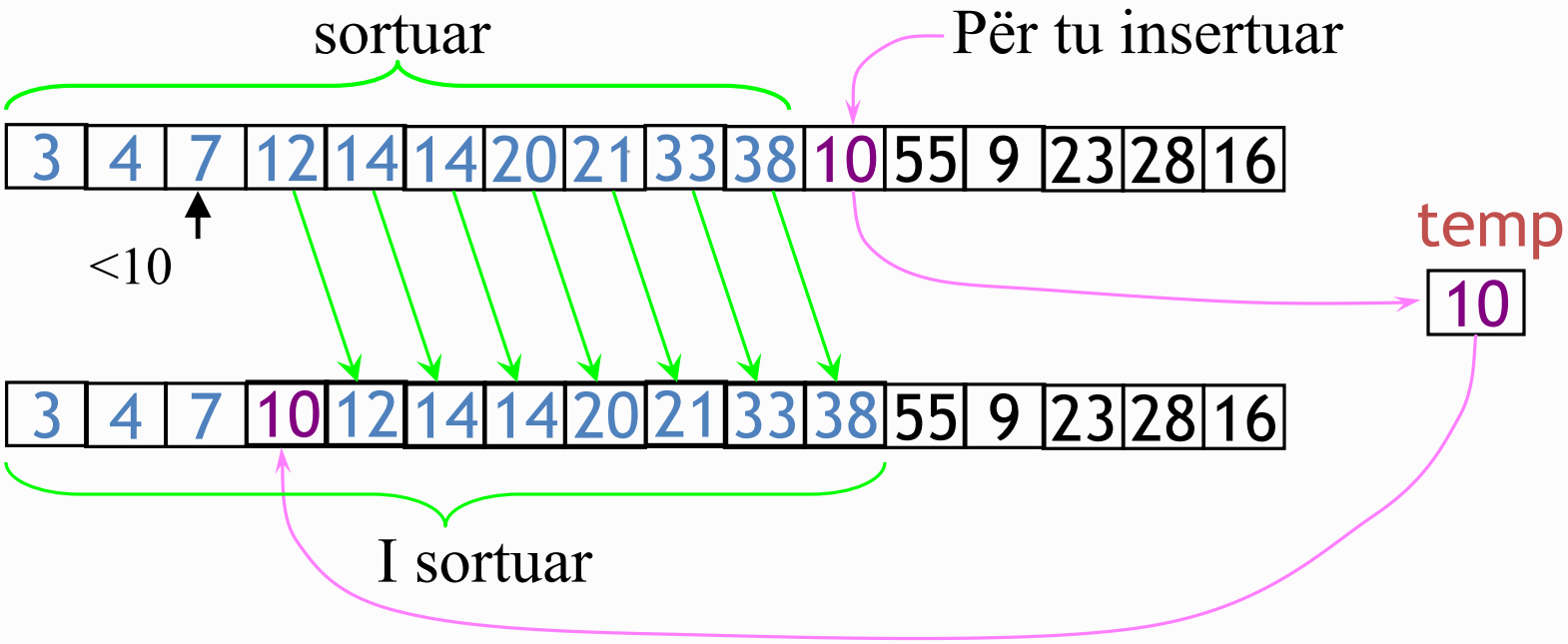
- Zakonisht krahasohet me lojën me letra.

- Sortimi Insertion ndan vargun në dy nën vargje
  - Nën vargu i majtë është i sortuar
  - Nën vargu i djathtë është i pasortuar



- Elementet në vargun e djathtë do të insertohen në nën vargun e majtë.

Sortimi Insertion



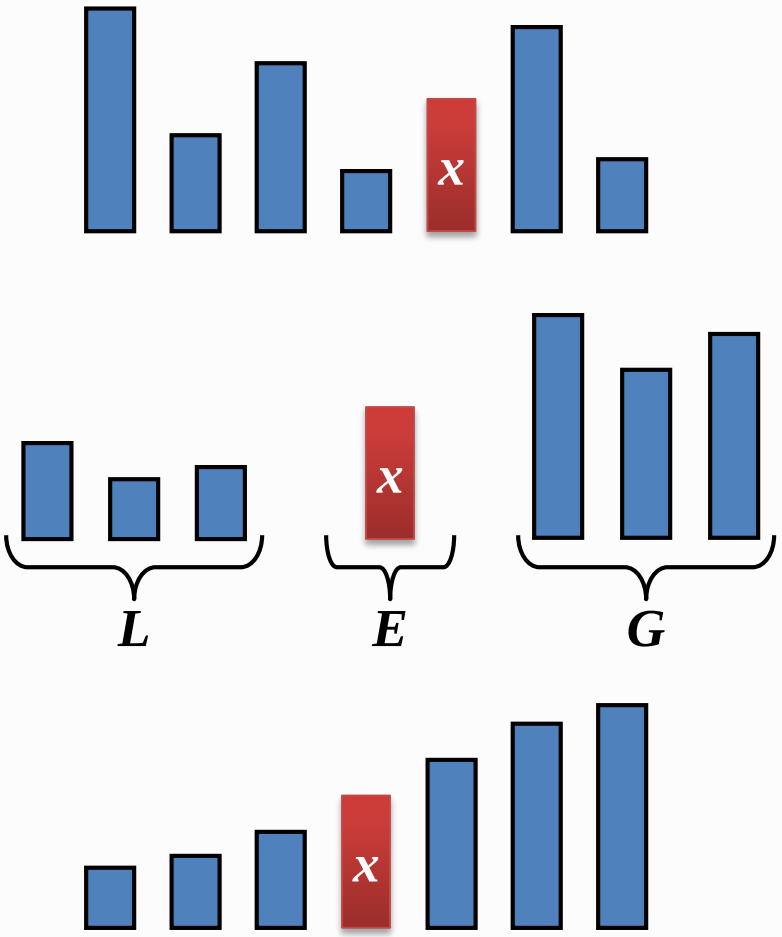
Shembull 13.3

- Të shkruhet programi për sortimin e një vargu duke përdorur algoritmin e sortimit **Insertion**. Sortimi të bëhet në rendin zbritës.

```
1  // Sortimi Insertion
2  #include <iostream>
3  using namespace std;
4
5  void InsertionSort(int vargu[], int m)
6  {
7      int i, j, insert;
8      for (j = 1; j < m; j++)
9      {
10         insert = vargu[j];
11         for (i = j - 1; (i >= 0) && (vargu[i] < insert); i--)
12         {
13             vargu[i + 1] = vargu[i];
14         }
15         vargu[i + 1] = insert;
16     }
17     return;
18 }
19
20 void shtyp(int vargu[], int m)
21 {
22     for (int i = 0; i < m; i++)
23         cout << vargu[i] << " ";
24     cout << "\n";
25 }
26
27 int main()
28 {
29     int vargu[] = { 5, 1, 12, -5, 16 };
30     cout << "Vargu origjinal: \n";
31     shtyp(vargu, 5);
32
33     InsertionSort(vargu, 5);
34     cout << "\nVargu i sortuar: \n";
35     shtyp(vargu, 5);
36     system("Pause");
37     return 0;
38 }
```

Sortimi Quick

- Sortimi Në rastin mesatar ka kompleksitetin O(n log n)
- Quick-sort është algoritëm i sortimit i bazur në logjikën "përçaj dhe sundo":
  - Përçarja: Përzgjidh një element x të rastësishëm (pivot) dhe particiono S në:
    - L elementet < x
    - E elementet = x
    - G elementet > x
  - Përseritja: sorto L dhe G
  - Sundimi: ngjit L, E dhe G



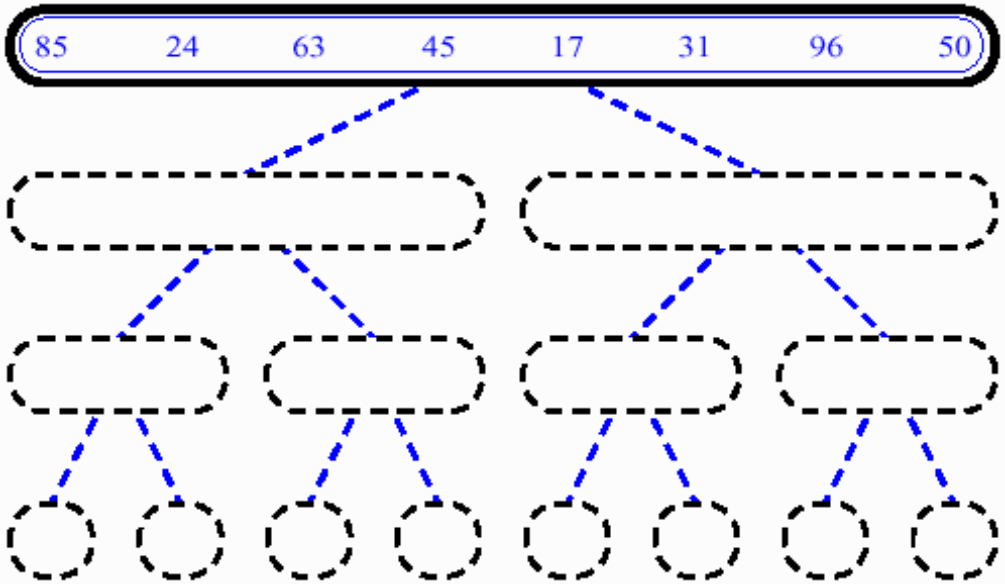
Shembull 13.4

- Të shkruhet programi për sortimin e një vargu duke përdorur algoritmin e sortimit **Quick**. Sortimi të bëhet në rendin zbritës.

```
1  // Soritmi Quick
2  #include <iostream>
3  using namespace std;
4
5  void shtyp(int vargu[], int m)
6  {
7      for (int i = 0; i < m; i++)
8          cout << vargu[i] << " ";
9      cout << "\n";
10 }
11
12 int copeto(int vargu[], int fillimi, int fundi)
13 {
14     int x = vargu[fillimi];
15     int i = fillimi - 1;
16     int j = fundi + 1;
17     int temp;
18     do
19     {
20         do
21         {
22             j --;
23         } while (x >vargu[j]);
24
25         do
26         {
27             i++;
28         } while (x <vargu[i]);
29
30         if (i < j)
31         {
32             temp = vargu[i];
33             vargu[i] = vargu[j];
34             vargu[j] = temp;
35         }
36     } while (i < j);
37     return j;
38 }
39
40 void QuickSort(int vargu[], int fillimi, int fundi)
41 {
42     int mesi;
43     if (fillimi < fundi)
44     {
45         mesi = copeto(vargu, fillimi, fundi);
46         QuickSort(vargu, fillimi, mesi);   // sort pjesn 1
47         QuickSort(vargu, mesi + 1, fundi);  // sorto pjesen 2
48     }
49     return;
50 }
51
52 int main()
53 {
54     int vargu[] = { 5, 1, 12, -5, 16 };
55     int n = sizeof(vargu) / sizeof(vargu[0]);
56     cout << "Vargu origjinal: \n";
57     shtyp(vargu, n);
58
59     QuickSort(vargu, 0, n - 1);
60
61     cout << "\nVargu i sortuar: \n";
62     shtyp(vargu, n);
63     system("Pause");
64     return 0;
65 }
```

Sortimi Merge

- Merge sort kombinon dy vargje të sortuara në një varg më të madh të sortuar.
- Algoritmi:
  - Ndahet vargu A[1…n] nË dy vargje B[1..n/2] dhe C[1..n/2]
  - Sorto vargun B dhe C
  - Bashko vargun e sortuar B dhe C në vargun A



Shembull 13.5

- Të shkruhet programi për sortimin e një vargu duke përdorur algoritmin e sortimit **Merge**.

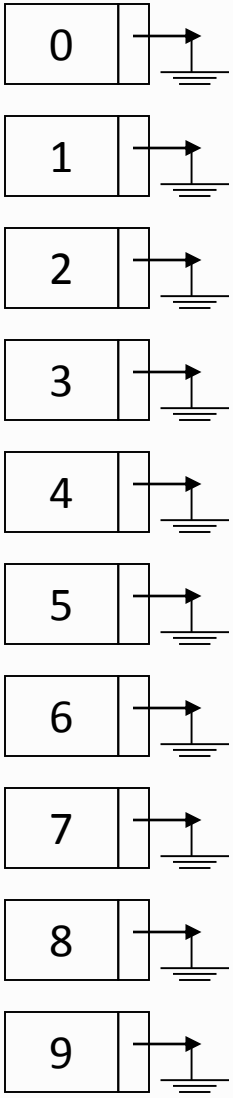
```cpp
// Soritmi Quick
#include <iostream>
using namespace std;

void shtyp(int vargu[], int m)
{
    for (int i = 0; i < m; i++)
        cout << vargu[i] << " ";
    cout << "\n";
}

void bashko(int vargu[], int m, int mes, int d)
{
    int i, j, k, n1 = mes - m + 1, n2 = d - mes;
    int M[5], D[5];// vargjet e perkohshme, majtas dhe djathtas

    for (i = 0; i < n1; i++)
        M[i] = vargu[m + i];
    for (j = 0; j < n2; j++)
        D[j] = vargu[mes + 1 + j];

    i = 0; j = 0; k = m;
    while (i < n1 && j < n2)    // Bashko vargjet
    {
        if (M[i] <= D[j])
        {
            vargu[k] = M[i];
            i++;
        }
        else
        {
            vargu[k] = D[j];
            j++;
        }
        k++;
    }

    while (i < n1)  // Kopjo elementet e mbetura të M[]
    {
        vargu[k] = M[i];
        i++;
        k++;
    }

    while (j < n2)      // Kopjo elementet e mbetura te D[]
    {
        vargu[k] = D[j];
        j++;
        k++;
    }
}

void mergeSort(int vargu[], int m, int d)
{
    if (m < d)
    {
        int mes = m + (d - m) / 2;
        mergeSort(vargu, m, mes);
        mergeSort(vargu, mes + 1, d);
        bashko(vargu, m, mes, d);
    }
}

int main()
{
    int vargu[] = { 5, 1, 12, -5, 16 };
    int n = sizeof(vargu) / sizeof(vargu[0]);
    cout << "Vargu origjinal: \n";

    shtyp(vargu, n);

    mergeSort(vargu, 0, n - 1);
    cout << "\nVargu i sortuar: \n";
    shtyp(vargu, n);

    system("Pause");
    return 0;
}
```

Sortimi Radix

- Radix sort konsideron strukturësn e vlerave kyçe për të bërë sortimin
- Algoritmi:
  - Sorto sipas shifrave më pak të rëndësishme
    - Numrat e njëjtë shkojnë në kovën e njëjtë
  - Ri rendit të gjithë numrat: numrat në kovën 1 i paraprijnë numrave në kovën 1, të cilët u paraprijnë numrave në kovën 2 dhe kështu me radhë…
  - Sorto sipas shifrave tjera më pak të rëndësishme
  - Vazhdo procesin gjersa numrat të jenë sortuar në të gjitha shifrat ***k***.

Shembull i sortimit Radix

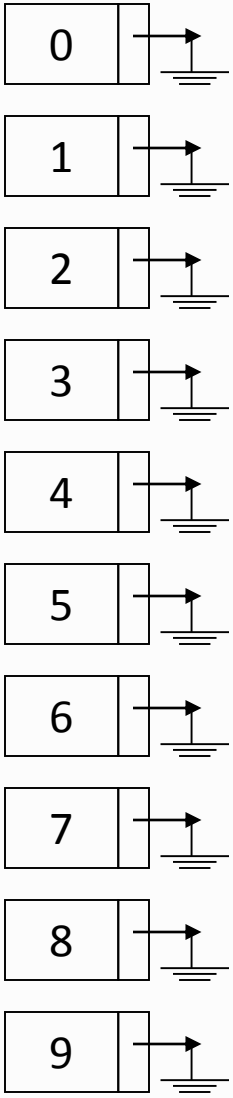
64	8	216	512	27	729	0	1	343	125



Kalimi 1

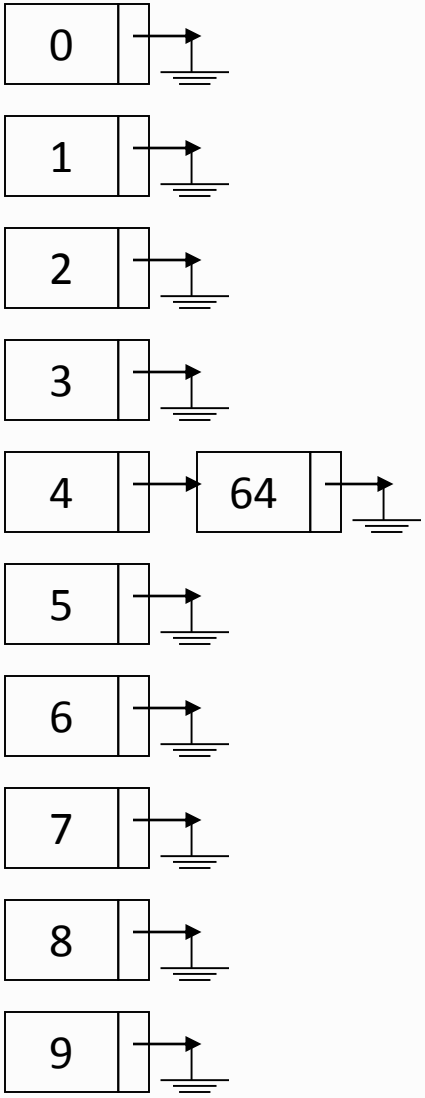
Shembull i sortimit Radix

64	8	216	512	27	729	0	1	343	125



Shembull i sortimit Radix

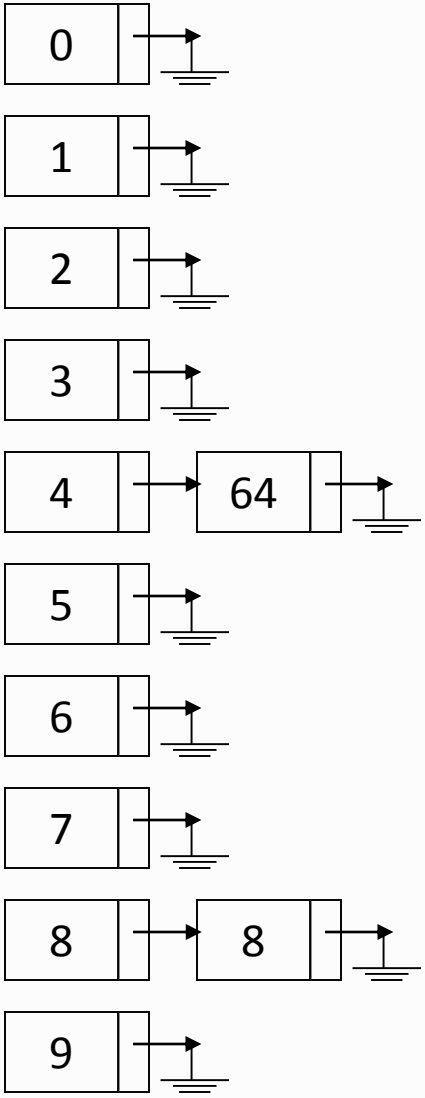
64	8	216	512	27	729	0	1	343	125



Kalimi 1

Shembull i sortimit Radix

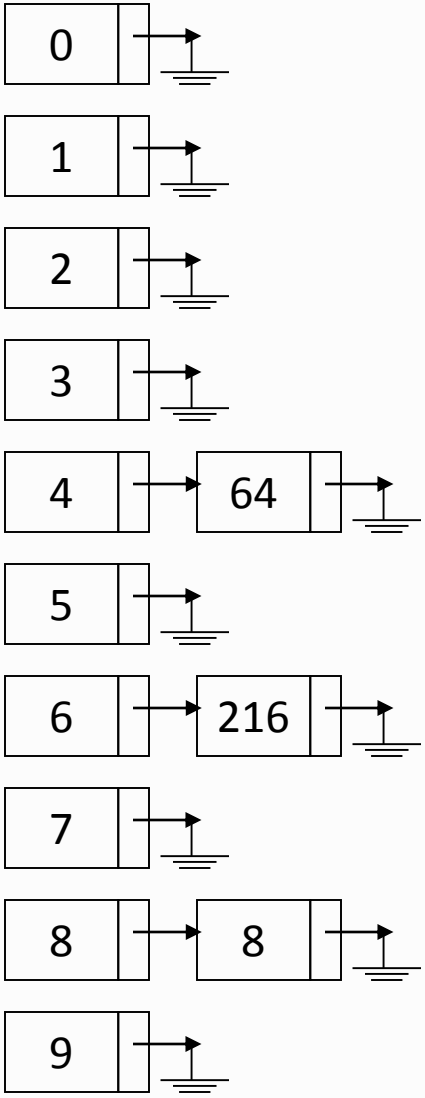
64	8	216	512	27	729	0	1	343	125



Kalimi 1

Shembull i sortimit Radix

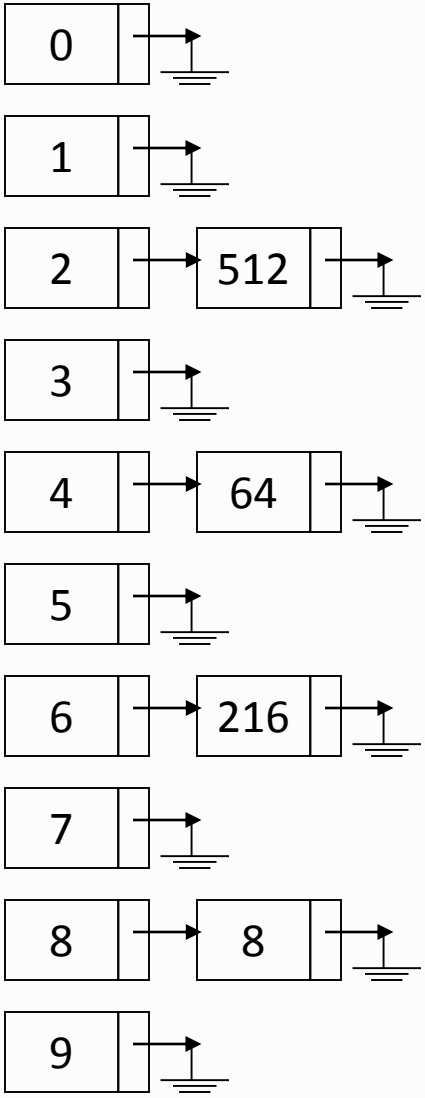
64	8	216	512	27	729	0	1	343	125



Kalimi 1

Shembull i sortimit Radix

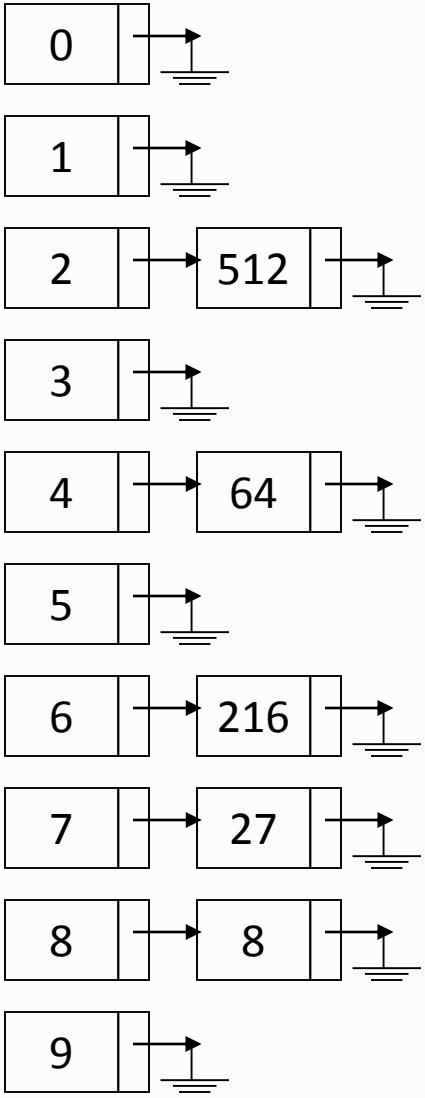
64	8	216	512	27	729	0	1	343	125



Kalimi 1

Shembull i sortimit Radix

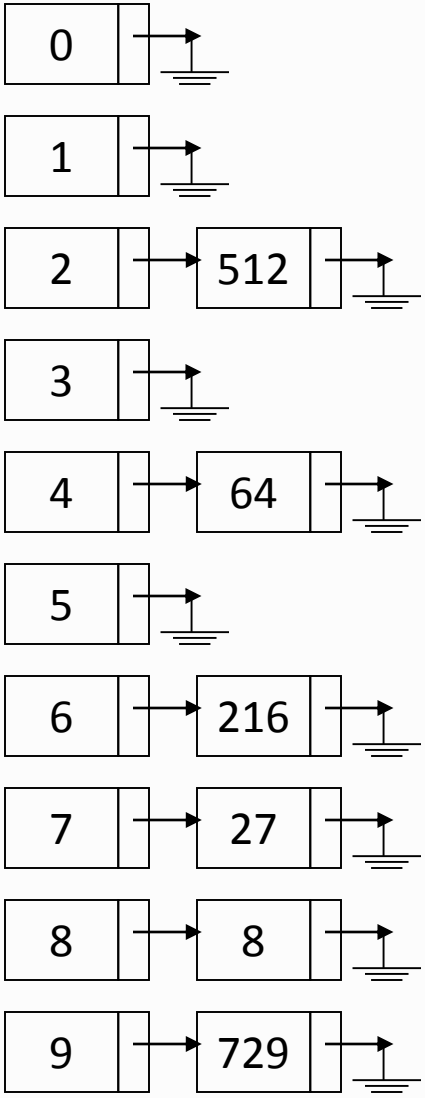
64	8	216	512	27	729	0	1	343	125



Kalimi 1

Shembull i sortimit Radix

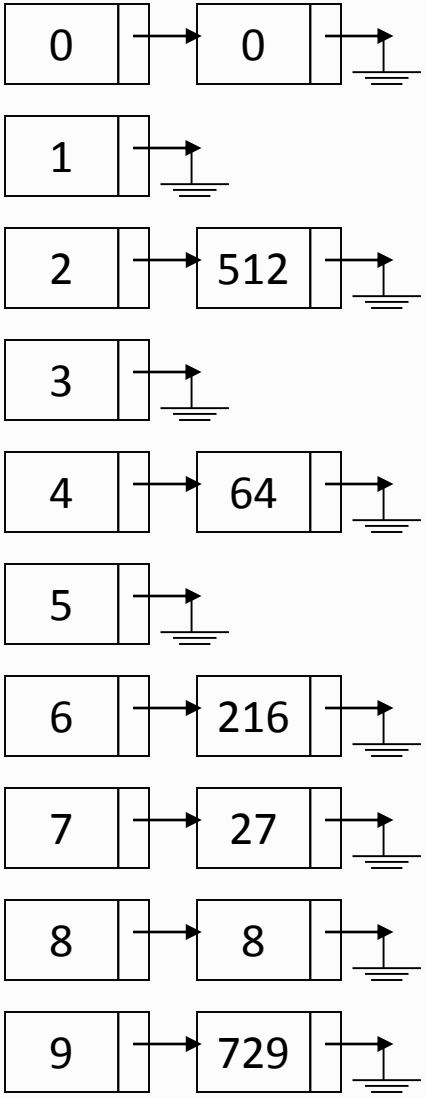
64	8	216	512	27	729	0	1	343	125



Kalimi 1

Shembull i sortimit Radix

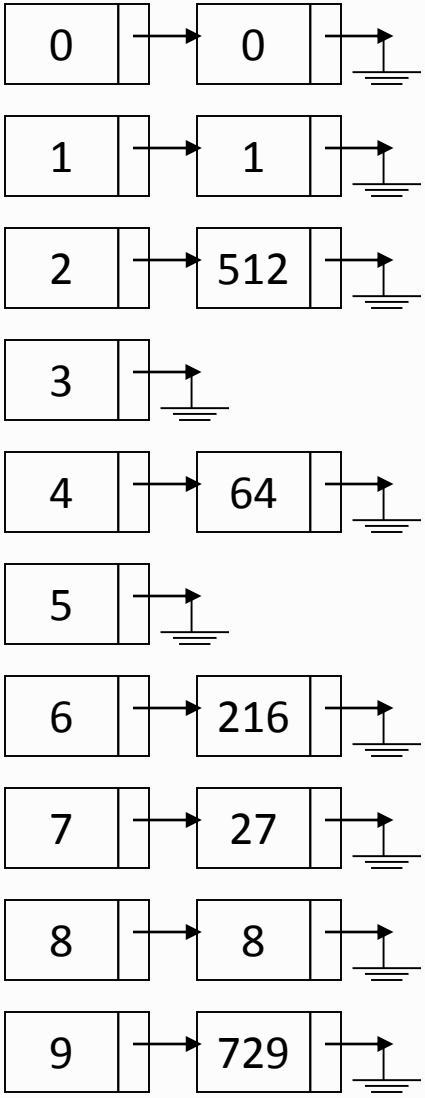
64	8	216	512	27	729	0	1	343	125



Kalimi 1

Shembull i sortimit Radix

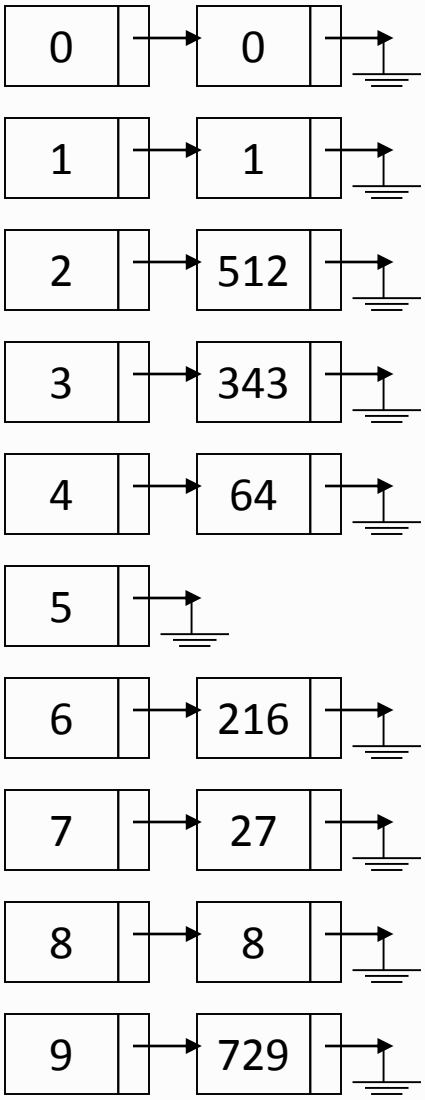
64	8	216	512	27	729	0	1	343	125



Kalimi 1

Shembull i sortimit Radix

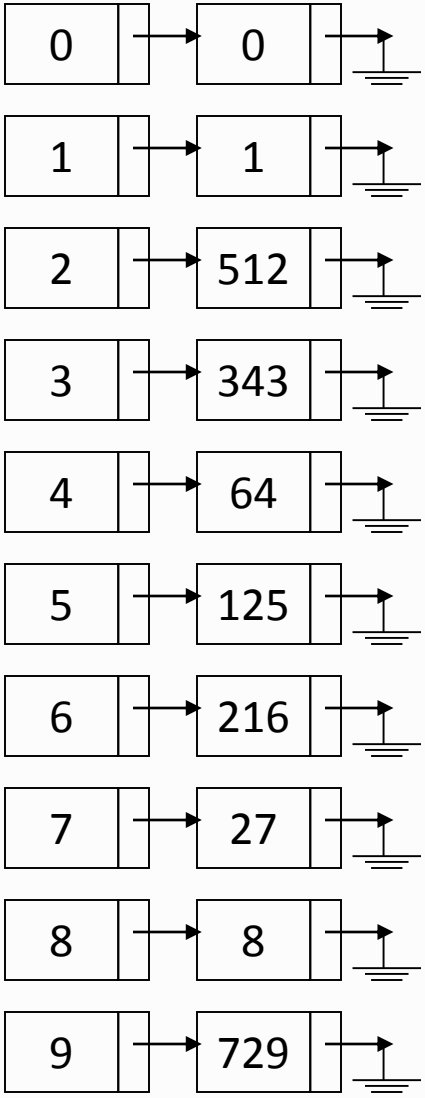
64	8	216	512	27	729	0	1	343	125



Kalimi 1

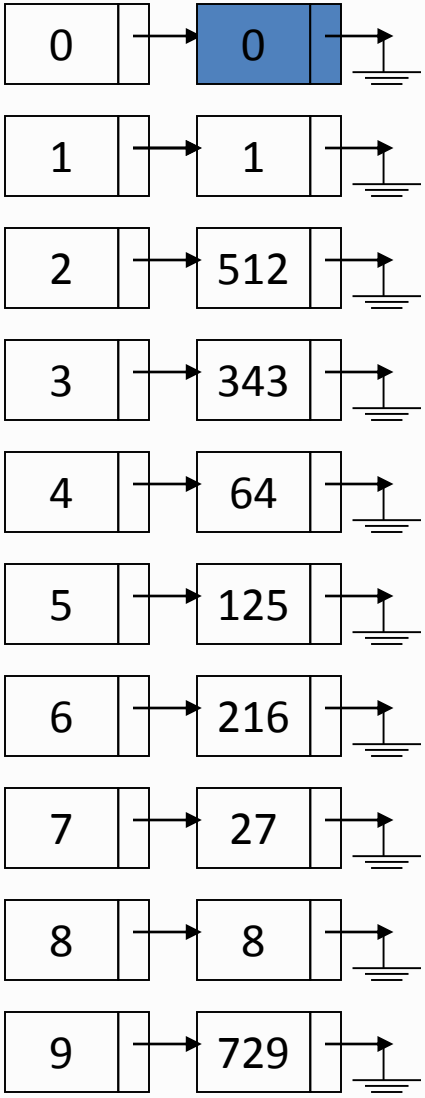
Shembull i sortimit Radix

64	8	216	512	27	729	0	1	343	125



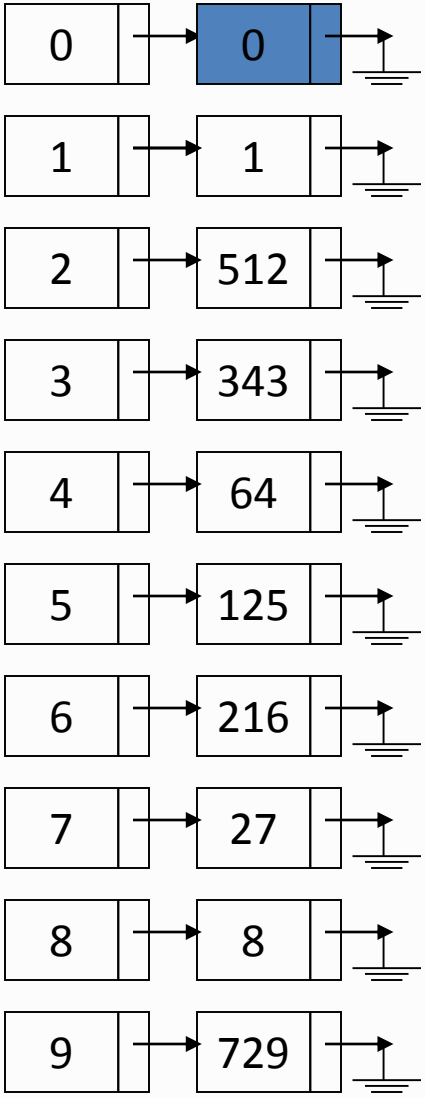
Kalimi 1

Shembull i sortimit Radix



Kalimi 1

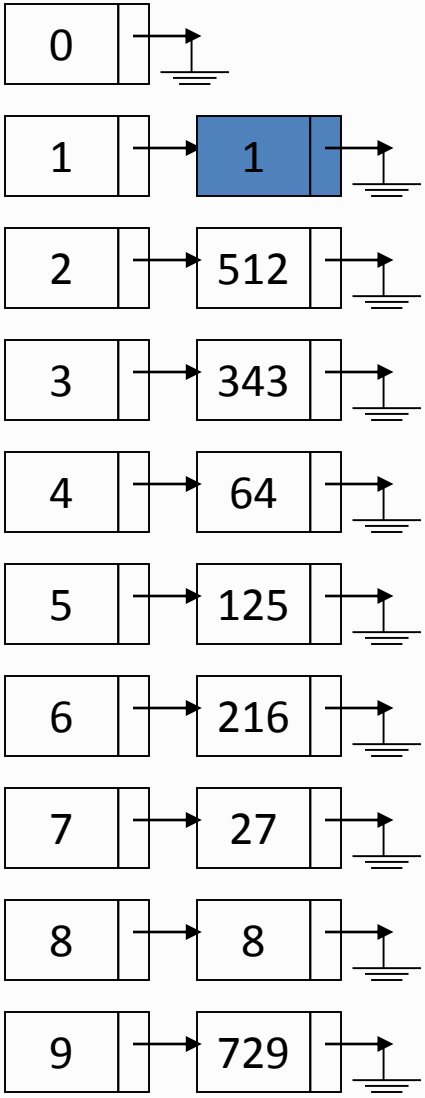
Shembull i sortimit Radix



Kalimi 1

Shembull i sortimit Radix

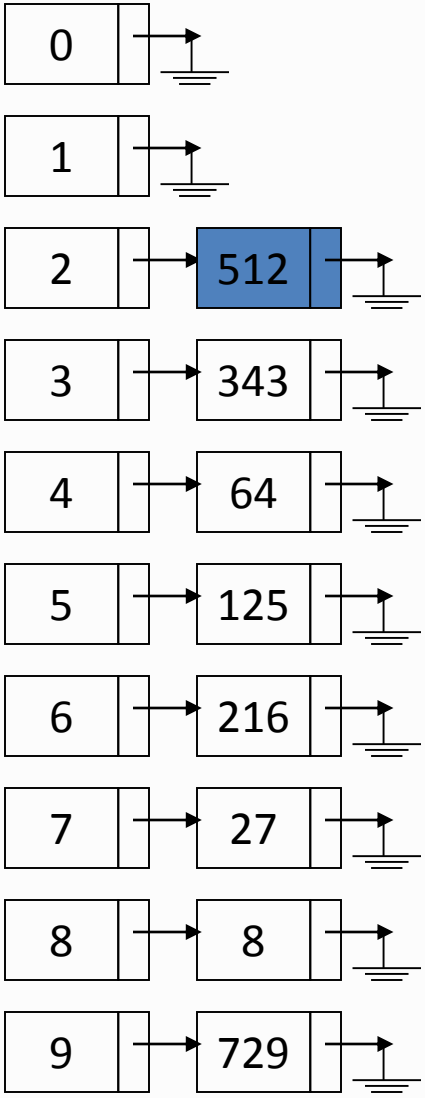
0	1								



Kalimi 1

Shembull i sortimit Radix

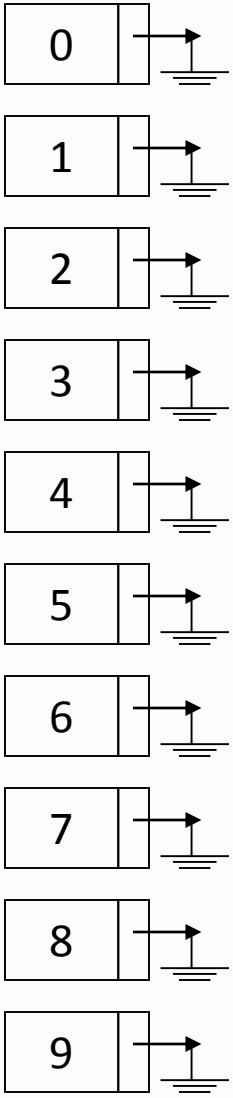
0	1	512							



Kalimi 1

Shembull i sortimit Radix

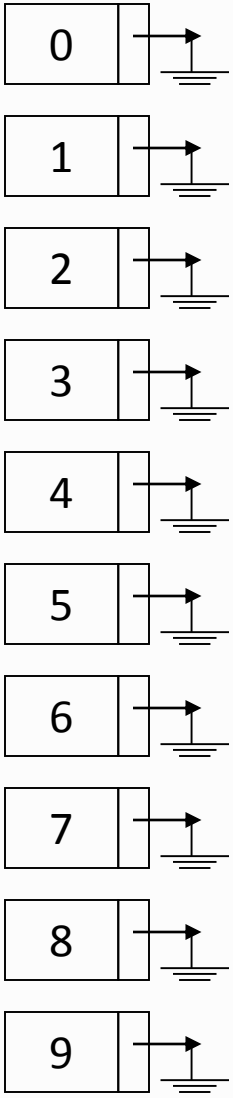
0	1	512	343	64	125	216	27	8	729



Kalimi 1

Shembull i sortimit Radix

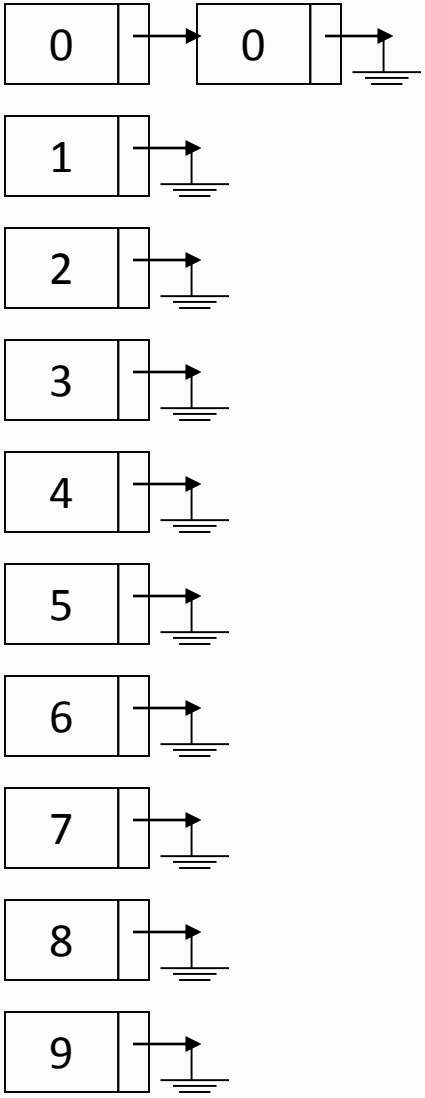
00	1	512	343	64	125	216	27	8	729



Kalimi 2

Shembull i sortimit Radix

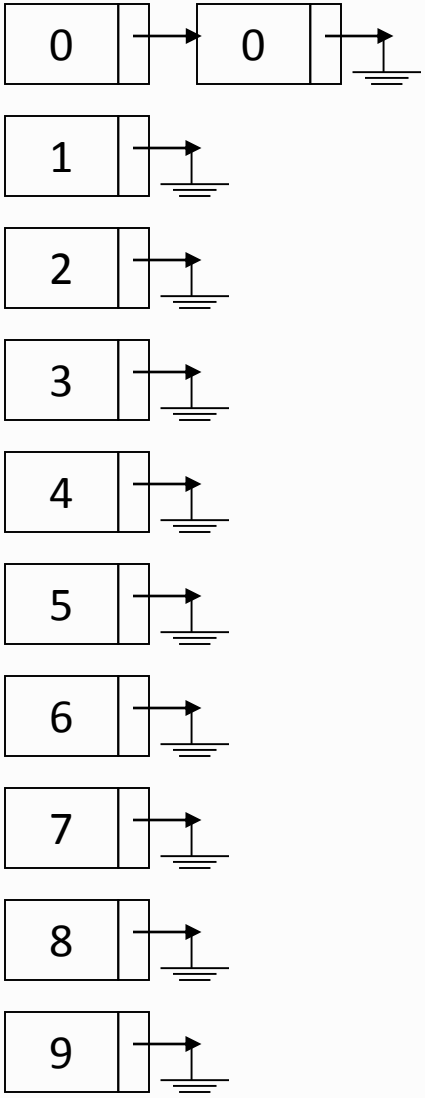
0	1	512	343	64	125	216	27	8	729



Kalimi 2

Shembull i sortimit Radix

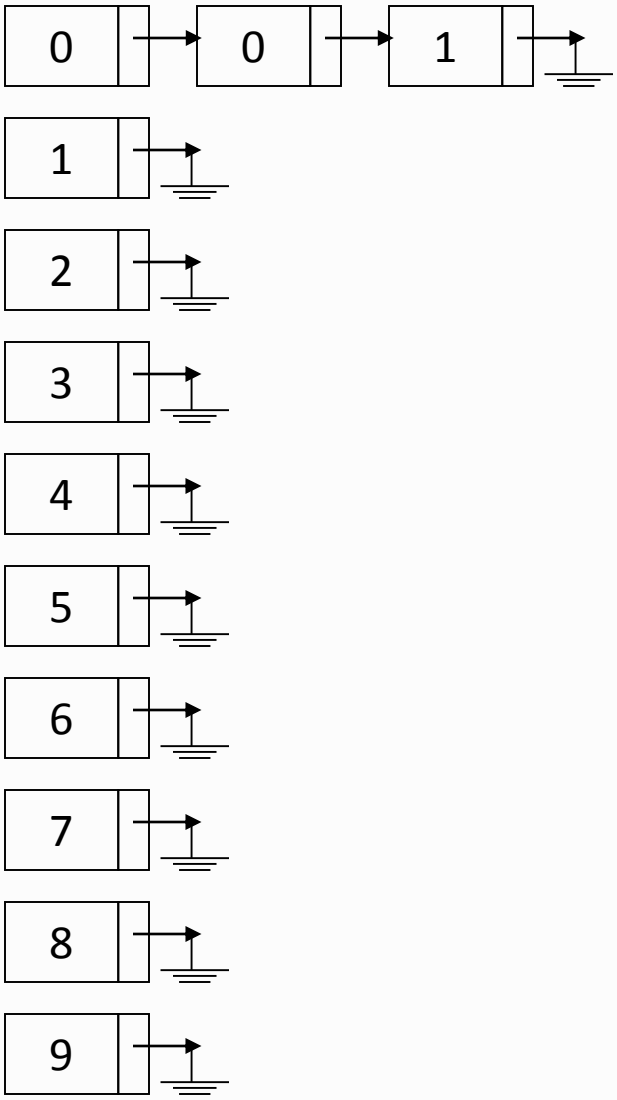
0	01	512	343	64	125	216	27	8	729



Kalimi 2

Shembull i sortimit Radix

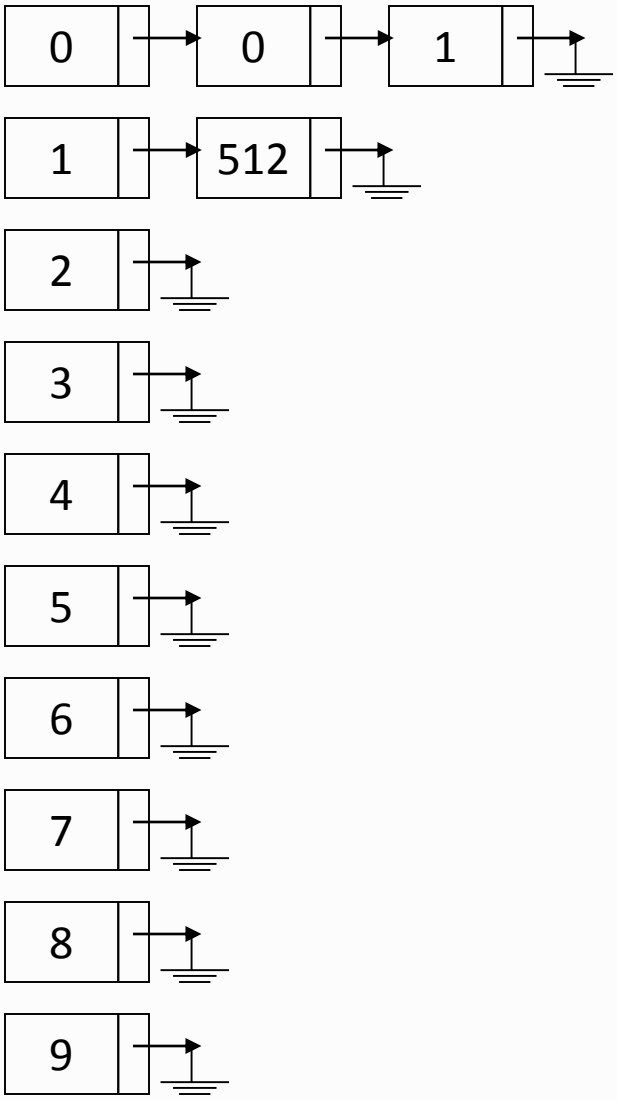
0	1	512	343	64	125	216	27	8	729



Kalimi 2

Shembull i sortimit Radix

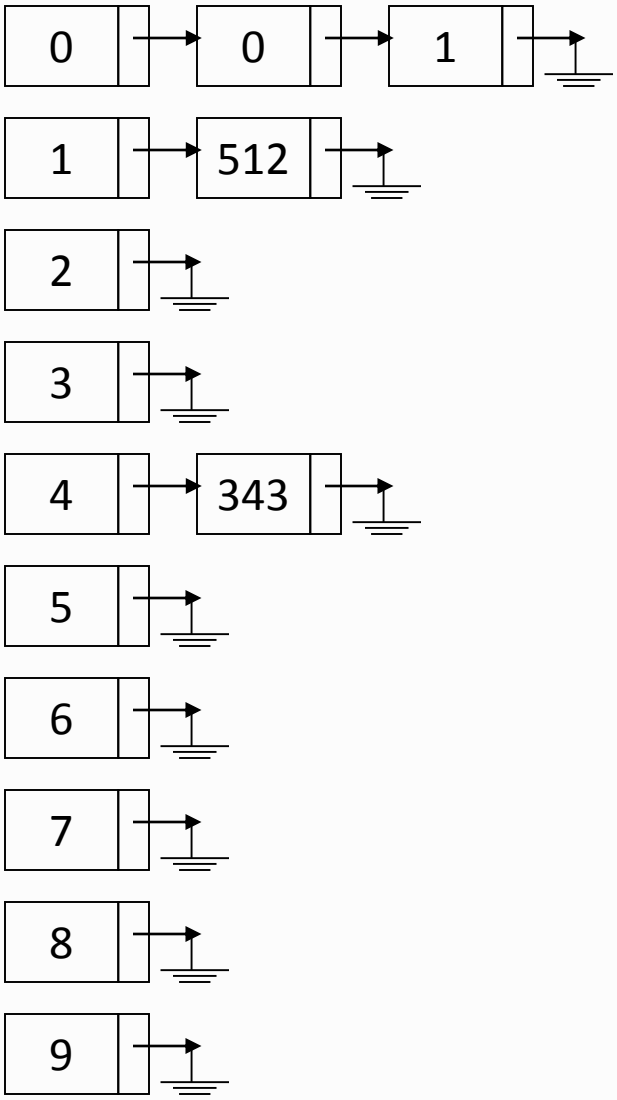
0	1	512	343	64	125	216	27	8	729



Kalimi 2

Shembull i sortimit Radix

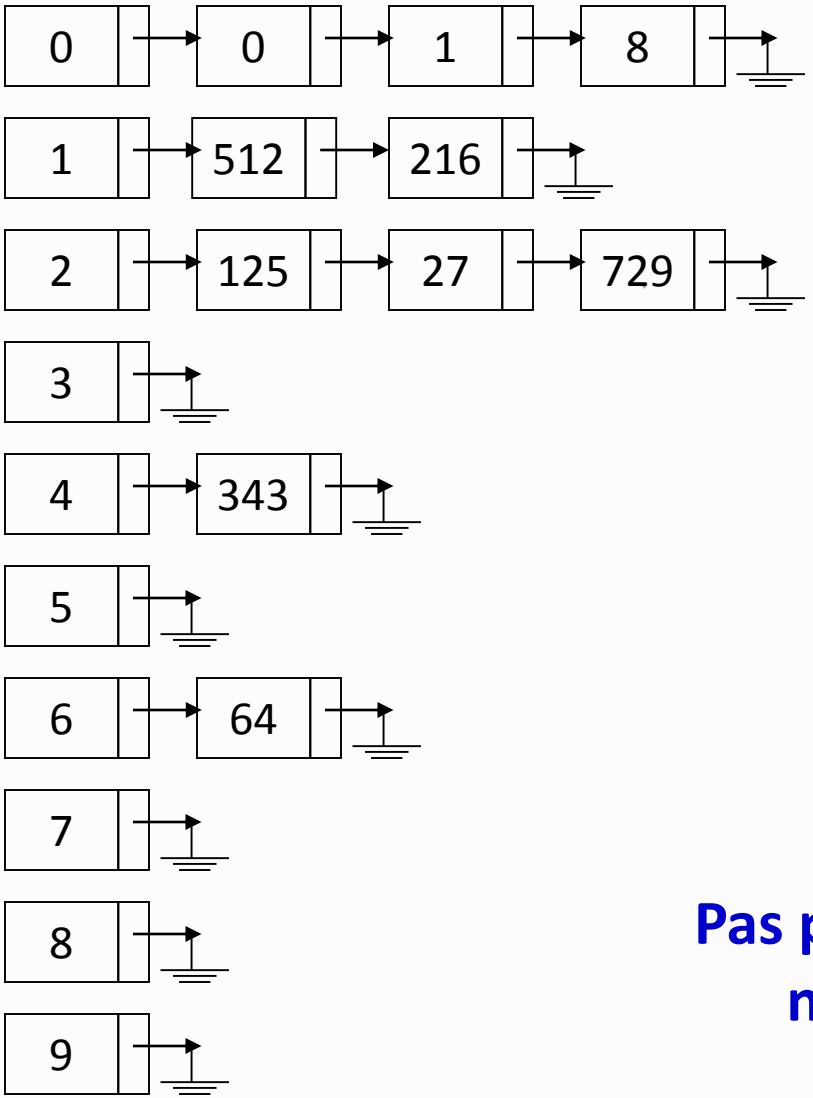
0	1	512	343	64	125	216	27	8	729



Kalimi 2

Shembull i sortimit Radix

0	1	512	343	64	125	216	27	8	729

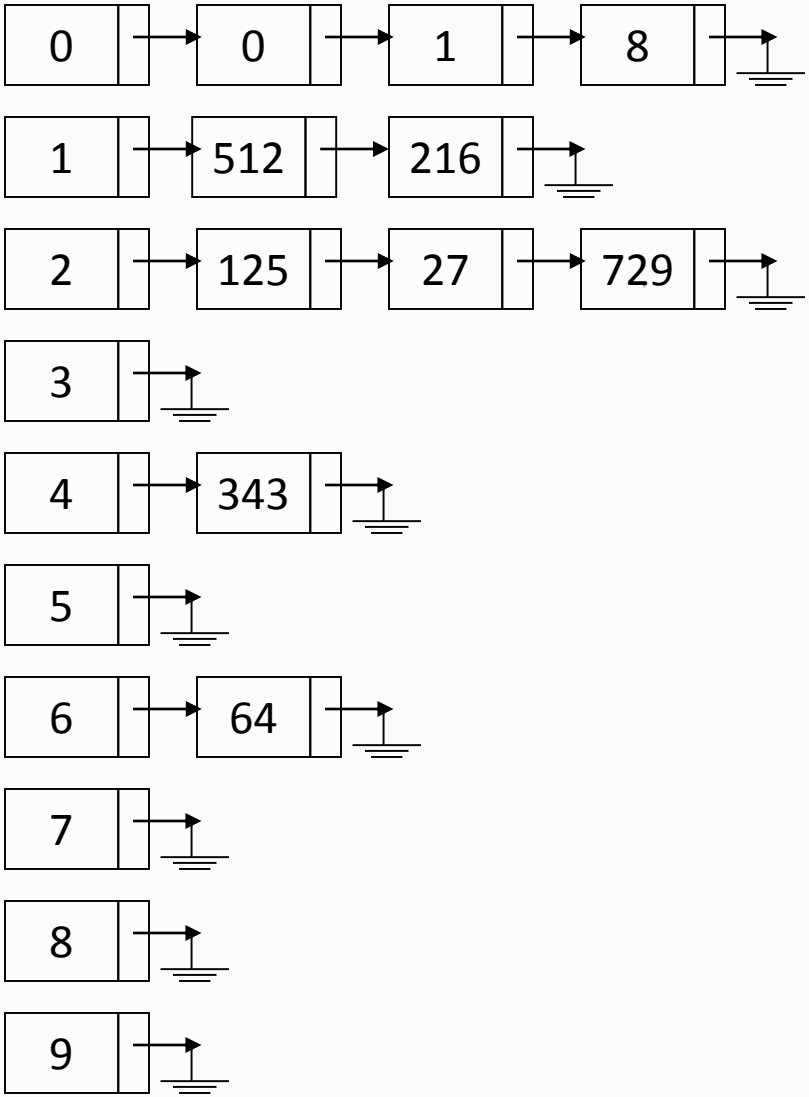


Kalimi 2

Pas përsëritjes së hapave deri tek numri 729 fitohet rezultati

Shembull i sortimit Radix

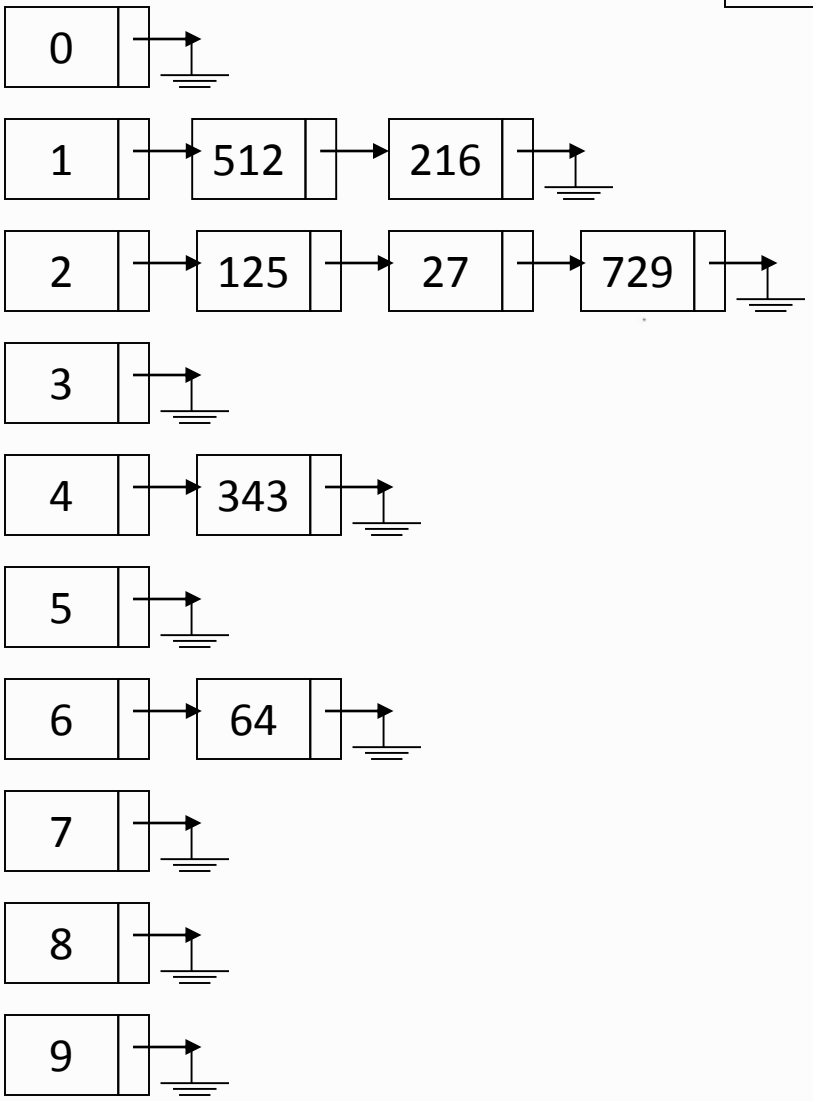
0	1	8							



Kalimi 2

Shembull i sortimit Radix

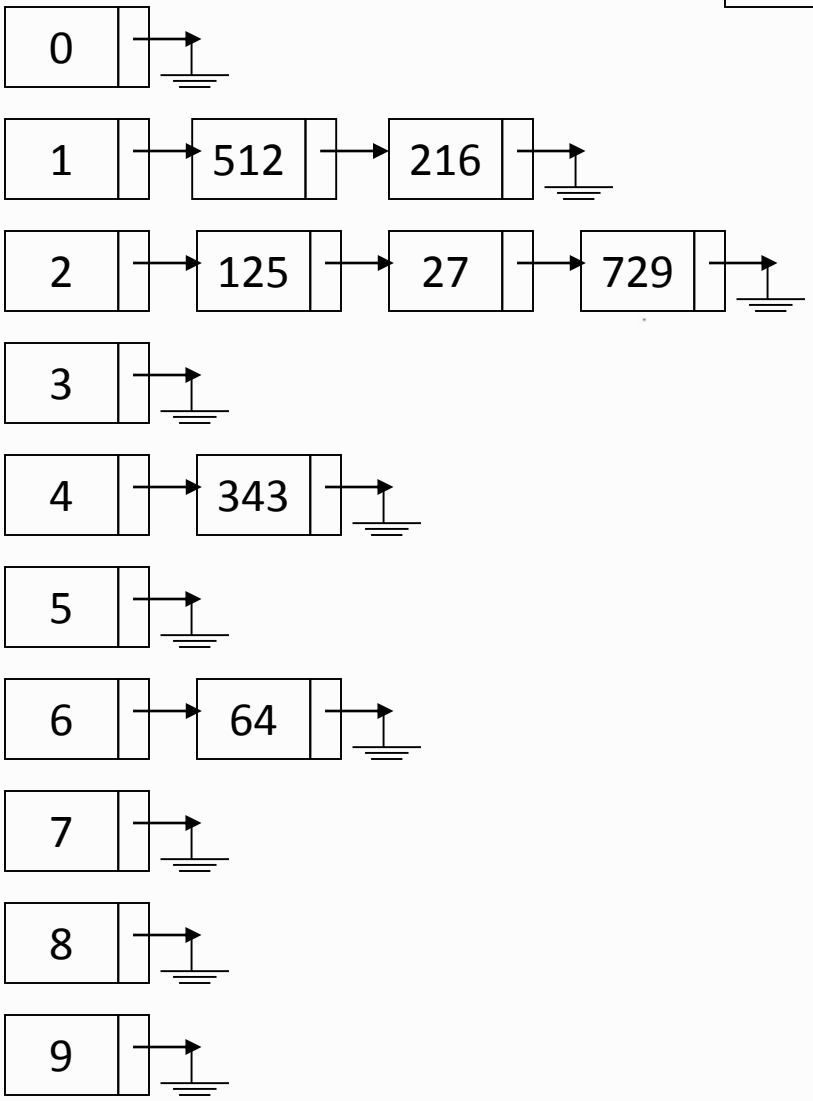
0	1	8							



Kalimi 2

Shembull i sortimit Radix

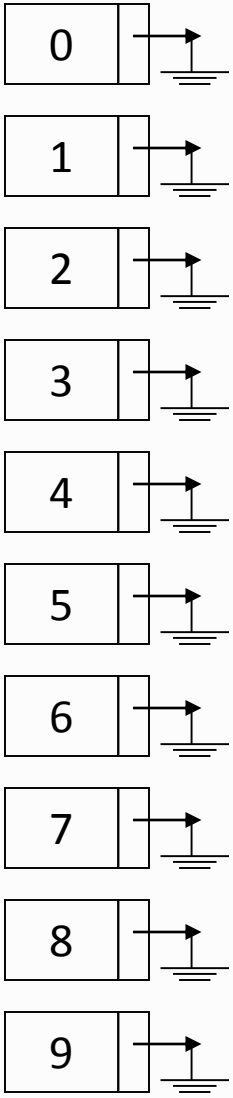
0	1	8	512	216					



Kalimi 2

Shembull i sortimit Radix

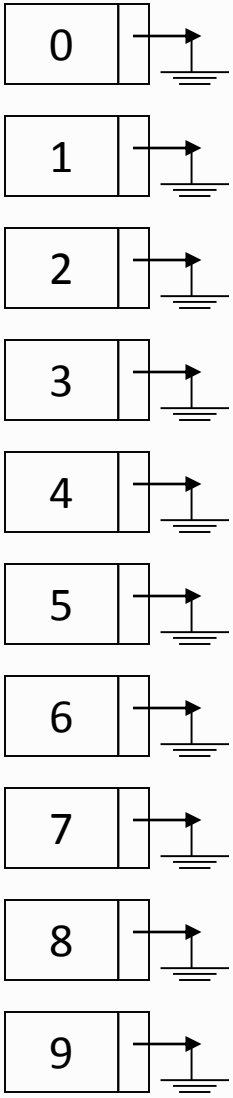
0	1	8	512	216	125	27	729	343	64



Fundi i kalimit 2

Shembull i sortimit Radix

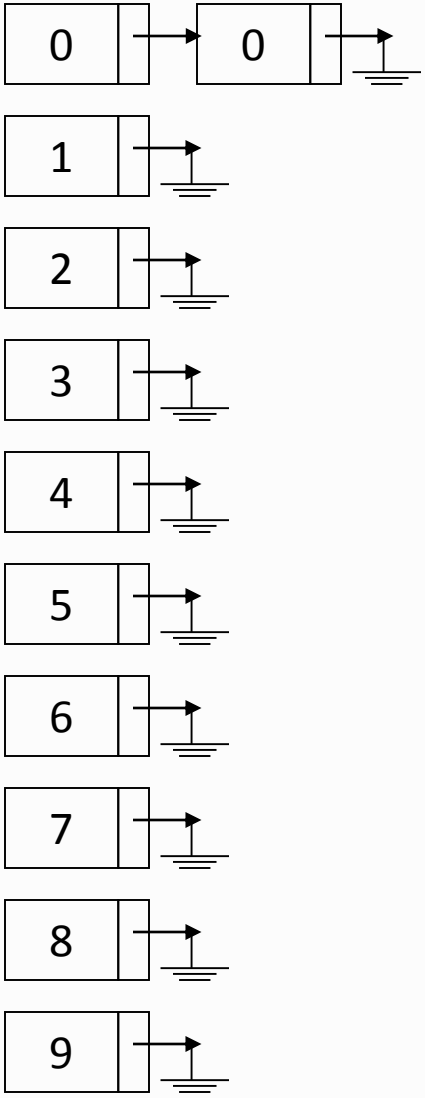
000	1	8	512	216	125	27	729	343	64



Kalimi 3

Shembull i sortimit Radix

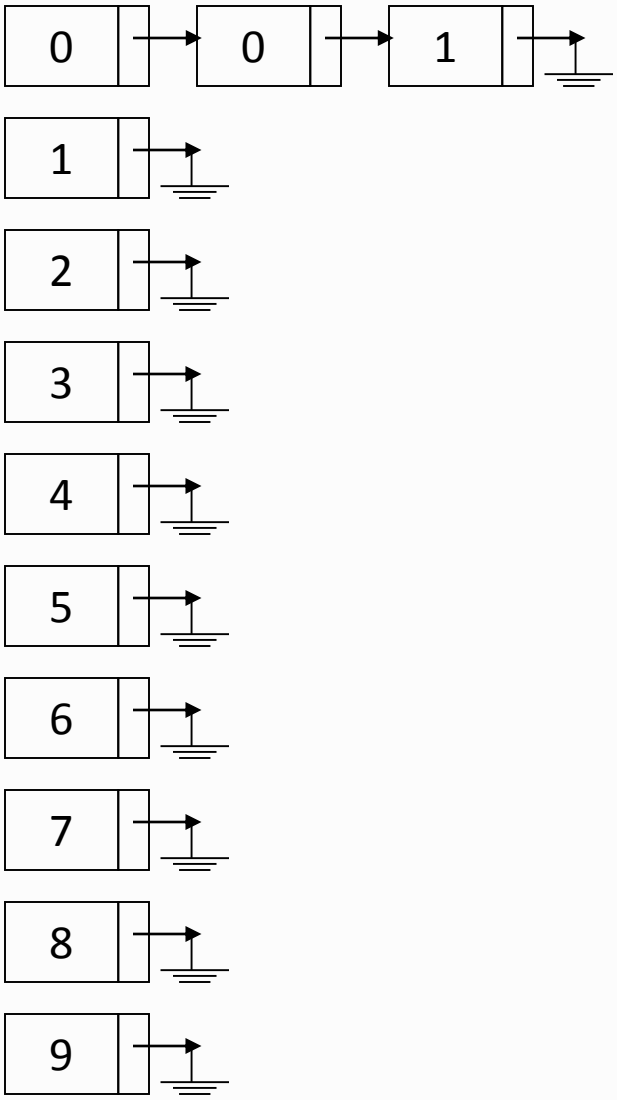
0	001	8	512	216	125	27	729	343	64



Kalimi 3

Shembull i sortimit Radix

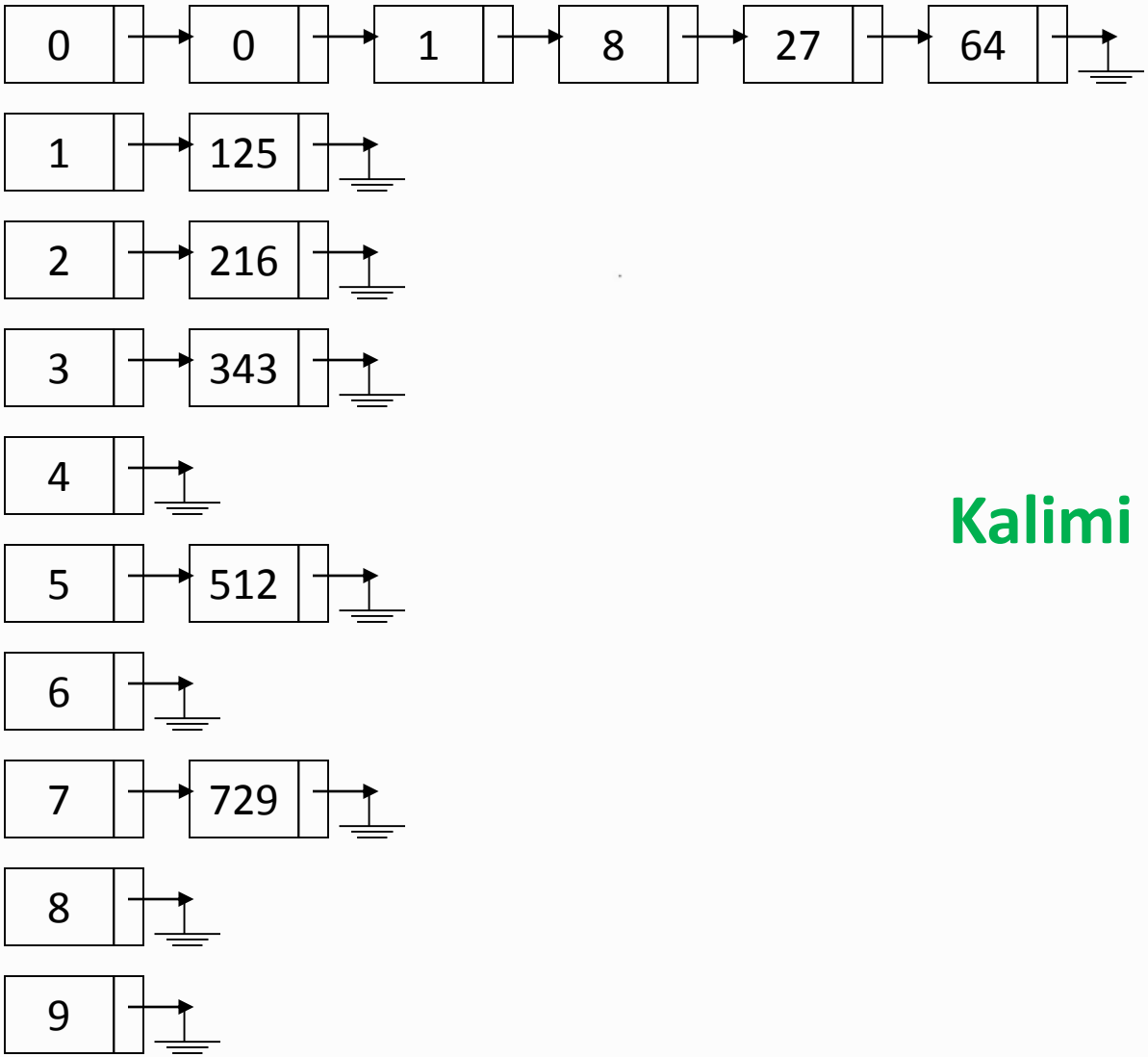
0	1	008	512	216	125	27	729	343	64



Kalimi 3

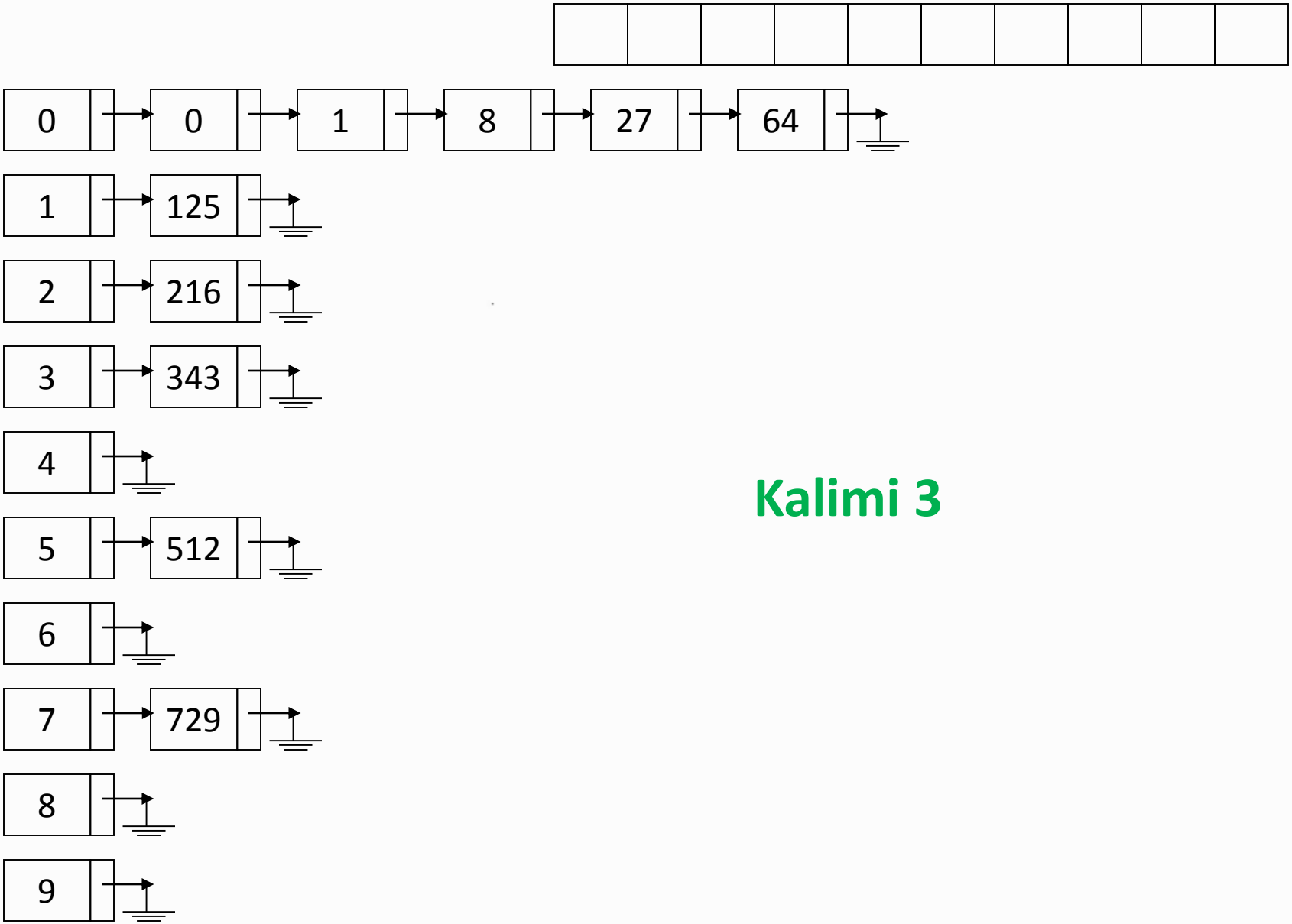
Shembull i sortimit Radix

0	1	8	512	216	125	27	729	343	64



Kalimi 3

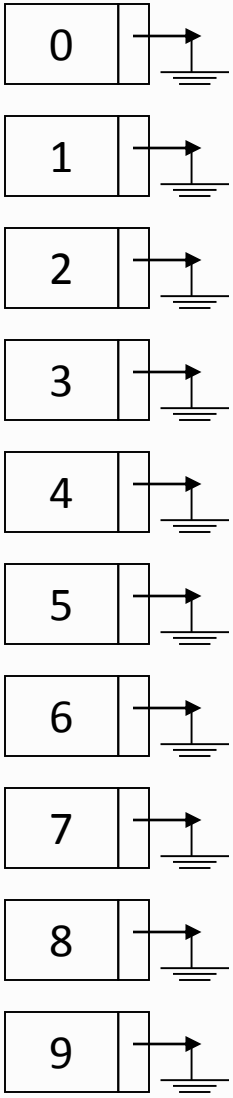
Shembull i sortimit Radix



Kalimi 3

Shembull i sortimit Radix

0	1	8	27	64	125	216	343	512	729



Fundi i kalimit 3

Shembull 13.6

- Të shkruhet programi për sortimin e një vargu duke përdorur algoritmin e sortimit **Radix**. Sortimi të bëhet në rendin zbritës.

```cpp
// Soritmi Radix
#include <iostream>
using namespace std;

void shtyp(int vargu[], int m)
{
    for (int i = 0; i < m; i++)
        cout << vargu[i] << " ";
    cout << "\n";
}

void radixSort(int *vargu, int n)
{
    int i;
    int nrMax = vargu[0];// Gjej max në varg për nderprerje të unazës
    for (i = 1; i < n; i++)
    {
        if (vargu[i] > nrMax)
            nrMax = vargu[i];
    }
    // ekzekuto unazën per secilen pozite decimale (duke pjestuar me modul të 10)
    //exp - eksponenti - pozita e shifres
    int exp = 1, kalimi = 0;
    int *tempKova = new int[n];
    while (nrMax / exp > 0)
    {
        int decimKova[10] = { 0 };
        // numëro rastet në këtë shifër decimale
        for (i = 0; i < n; i++)
            decimKova[vargu[i] / exp % 10]++;

        // Përgatit numratorët e pozitave që do të përdoren për ri-renditjen
        // e vlerave per kete pozite decimale
        for (i = 1; i < 10; i++)
            decimKova[i] += decimKova[i - 1];

        // Ri-renditi numrat ne tmpbuffer dhe pastaj kopjoj ne buffer-in origjinal
        // Per ta mbajtur renditjen, fillo prej fundit
        for (i = n - 1; i >= 0; i--)
            tempKova[--decimKova[vargu[i] / exp % 10]] = vargu[i];
        for (i = 0; i < n; i++)
            vargu[i] = tempKova[i];

        exp *= 10;      // Kalo në poziten e ardhshme decimale.
        kalimi = kalimi + 1;
        cout << "\nKalimi " << kalimi << " : ";
        shtyp(vargu, n);
    }
}

int main()
{
    const int N = 8;

    int vargu[N] = { 310, 213, 23, 130, 13, 301, 222, 89 };
    cout << "Hyrja    : ";
    shtyp(vargu, N);
    cout << endl;

    radixSort(vargu, N);

    cout << "\n\nDalja    : ";
    shtyp(vargu, N);
    cout << "\n";

    system("Pause");
    return 0;
}
```

Sortimi Shell

- Është zhvilluar nga Donald Shell më 1959.

- Në vend të krahasimit të elementeve fqinje, shell sort i krahason elementet që janë të një distancë të caktuar larg nga njëri-tjetri (d përfaqëson këtë distancë).

- Vlera e d (d = (N + 1) / 2) fillon sa gjysma e madhësisë hyrëse dhe përgjysmohet pas çdo kalimi nëpër varg.
-
- Elementet krahasohen dhe shkëmbehen kur ka nevojë. të e nevojshme.

Shembull 13.7

- Të shkruhet programi për sortimin e një vargu duke përdorur algoritmin e sortimit **Shell**.

```
1  // Sortimi Shell
2  #include <iostream>
3  using namespace std;
4
5  void shtyp(int vargu[], int m)
6  {
7      for (int i = 0; i < m; i++)
8          cout << vargu[i] << " ";
9      cout << "\n";
10 }
11
12 int shellSort(int vargu[], int n)
13 {
14     int dist;
15     for (dist = n / 2; dist > 0; dist = dist / 2)
16     {
17         for (int i = dist; i < n; i ++)
18         {
19             int temp = vargu[i];
20             int j;
21             for (j = i; j >= dist && vargu[j - dist] > temp; j = j-dist)
22                 vargu[j] = vargu[j - dist];
23
24             vargu[j] = temp;
25         }
26     }
27     return 0;
28 }
30 int main()
31 {
32     int vargu[] = { 5, 1, 12, -5, 16 };
33     int n = sizeof(vargu) / sizeof(vargu[0]);
34     cout << "Vargu fillestar: \n";
35     shtyp(vargu, n);
36
37     shellSort(vargu, n);
38
39     cout << "\nVargu i sortuar: \n";
40     shtyp(vargu, n);
41     system("Pause");
42     return 0;
43 }
```