«ԾՐԱԳՐԱՎՈՐՄԱՆ ՀԻՄՈՒՆՔՆԵՐ» դասընթաց

այլ ոլորտներից դեպի տեխնոլոգիական ոլորտ սկսնակների համար

edu2020.am

ԴԱՍ #8

ՀԱՅ-ՌՈՒՍԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

ՀԱՅԱՍՏԱՆԻ ՀԱՆՐԱՊԵՏՈՒԹՅԱՆ
ԲԱՐՁՐ ՏԵԽՆՈԼՈԳԻԱԿԱՆ
ԱՐԴՅՈՒՆԱԲԵՐՈՒԹՅԱՆ ՆԱԽԱՐԱՐՈՒԹՅՈՒՆ

Զանգվածների սորտավորում

int arr [7]

44	21	26	6	192	5	-6

Սորտավորել զանգվածը աճման կարգով (դասավորել փոքրից մեծ)

-6	5	6	21	26	44	192

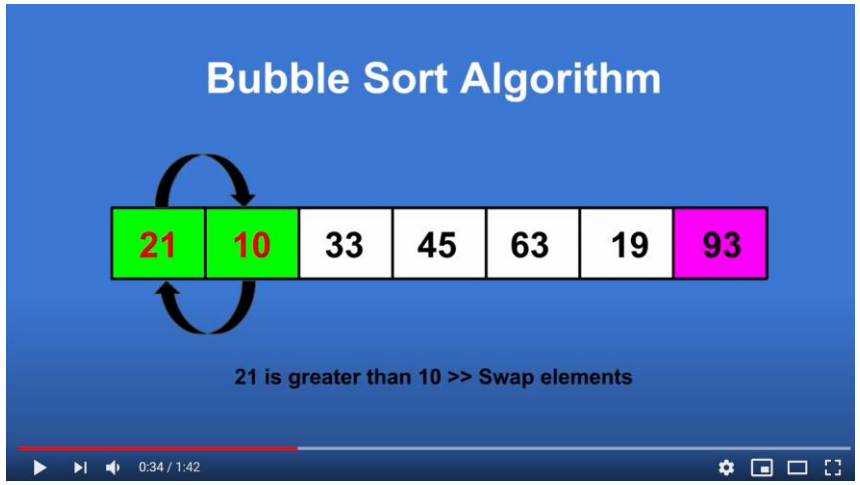
Սորտավորել զանգվածը նվազման կարգով (դասավորել մեծից փոքր)

192	44	26	21	6	5	-6

Ալգորիթմ 1. Պղպջակի մեթոդ

Ալգորիթմի քայլերը (տրտավորել աճման կարգով).

1. Հերթով համեմատել երկու հարևան էլեմենտները: Եթե առաջին էլեմենտը մեծ է երկրորդից, տեղափոխել դրանք:
2. Կատարել 1 գործողությունները այնքան ժամանակ քանի դեռ տեղի է ունենում զանգվածի էլեմենտների տեղափոխություն

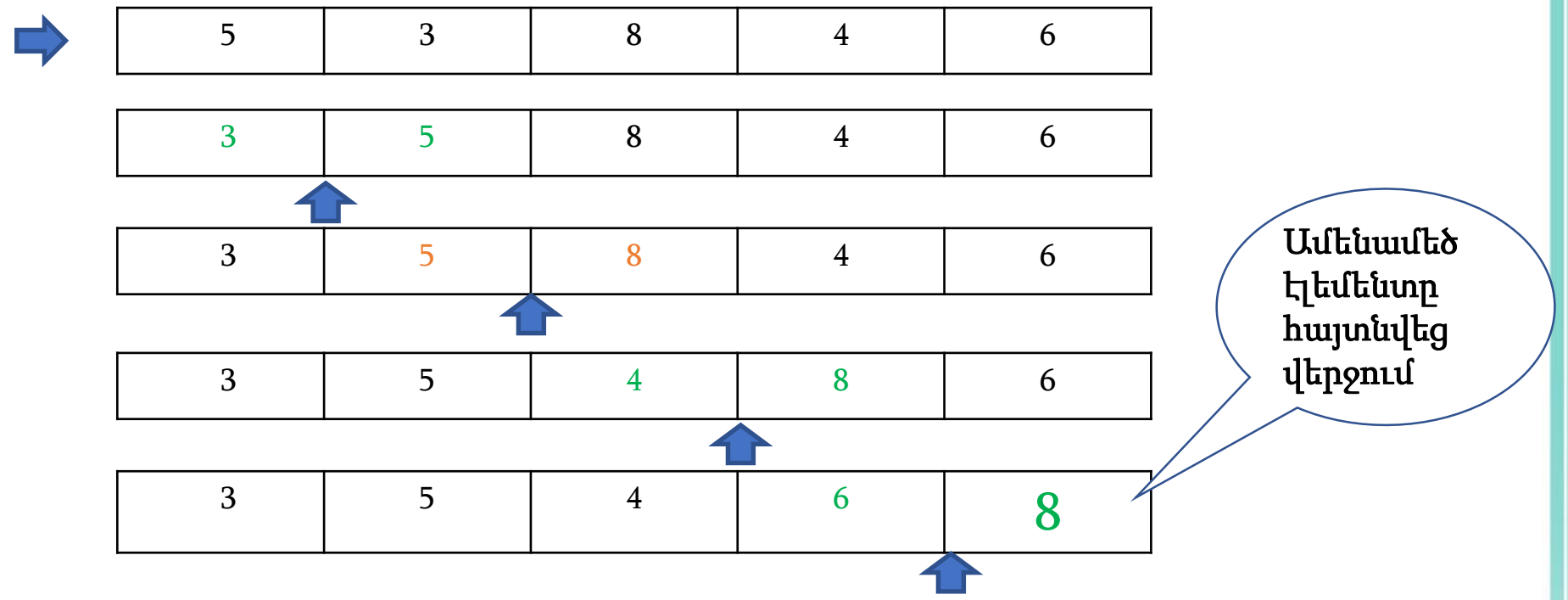


https://www.youtube.com/watch?v=Ex_BMuUcjkc

Ալգորիթմ 1. Պղպջակի մեթոդ, օրինակ

Մուտք՝ 5 չափանի ստատիկ զանգված

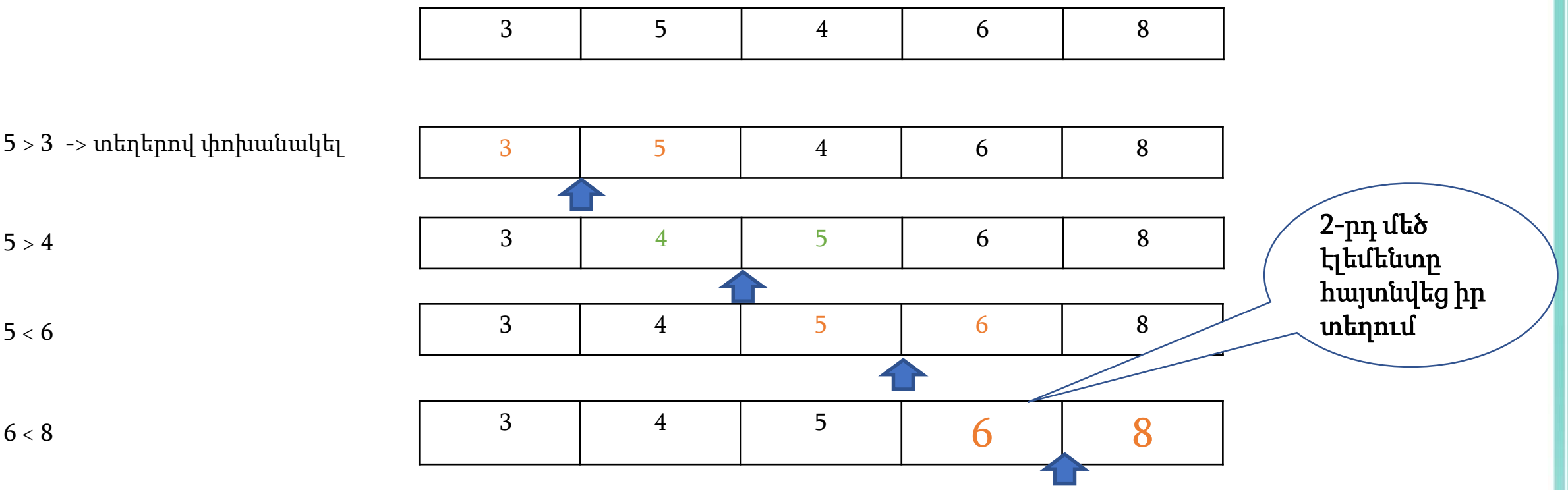
Ելք՝ Զանգված սորտավորված աճման կարգով



Կրկնել նույն քայլերը, քանի որ տեղի է ունեցել տեղերի փոխանակում

Ալգորիթմ 1. Պղպջակի մեթոդ, օրինակ

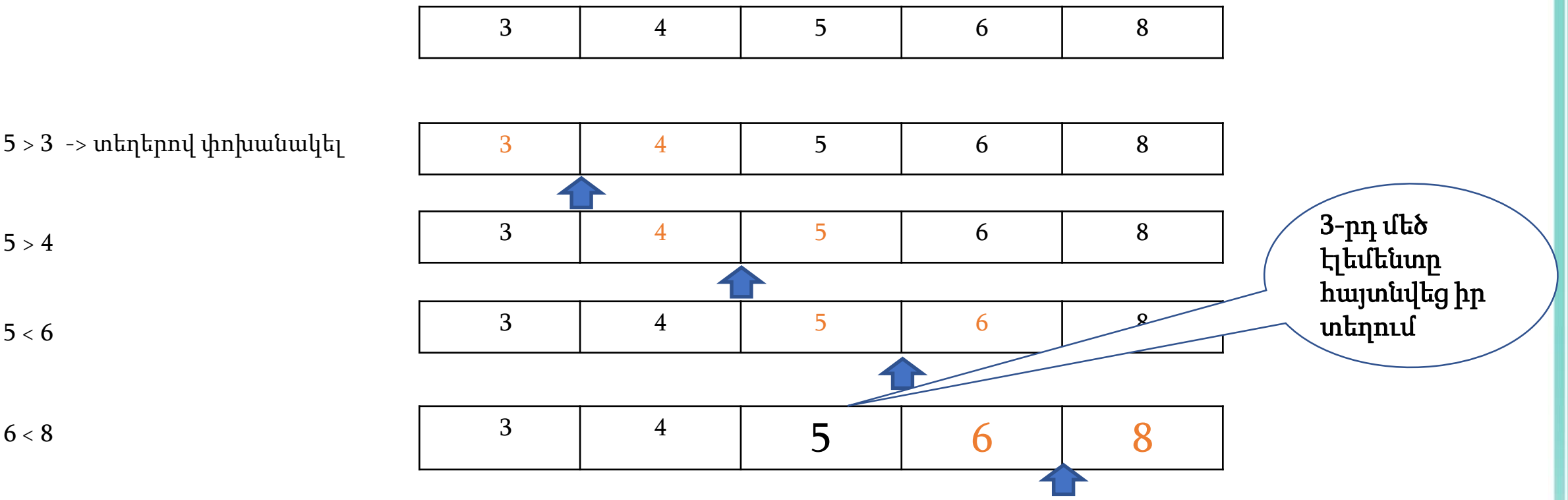
Նորից սկսում ենք համեմատել զանգվածի սկզբից



Կրկնել նույն քայլերը, քանի որ տեղի է ունեցել տեղերի փոխանակում

Ալգորիթմ 1. Պղպջակի մեթոդ, օրինակ

Նորից սկսում ենք համեմատել զանգվածի սկզբից



Ավարտել, քանի որ տեղերի փոխանակում տեղի չի ունեցել

Ալգորիթմի իրականացում 1

```
#include <iostream>
int main() {
   const int size = 5;
   int arr[] = {5, 3, 8, 4, 6};

   bool swapped = false;
   do {
     swapped = false;
     for (int i = 0; i < size - 1; i++) {
         if (arr[i] > arr[i + 1]) {
            int temp = arr[i];
            arr[i] = arr[i+1];
            arr[i+1] = temp;
            swapped = true;
        }
     }
   } while (swapped);

    for (int i = 0; i < size; i++) {
        std::cout << arr[i]<< " ";
    }
}
```

Հաշվի չենք առնում որ վերջին էլեմենտները արդեն սորտավորված են

https://repl.it/@VahagVardanyan/BubleSort

Ալգորիթմի իրականացում 1

```
#include <iostream>
int main() {
   const int size = 5;
   int arr[] = {5, 3, 8, 4, 6};

   bool swapped = false;
   do {
     swapped = false;
     for (int i = 0; i < size - 1; i++) {
         if (arr[i] > arr[i + 1]) {
            int temp = arr[i];
            arr[i] = arr[i+1];
            arr[i+1] = temp;
            swapped = true;
        }
     }
   } while (swapped);

    for (int i = 0; i < size; i++) {
        std::cout << arr[i]<< " ";
    }
}
```

Հաշվի չենք առնում որ վերջին էլեմենտները արդեն սորտավորված են

Այստեղ կարելի է տպել զանգվածը, քայլերի հերթականությունը ավելի լավ հասկանալու համար

https://repl.it/@VahagVardanyan/BubleSort

Ալգորիթմի իրականացում 2

```
#include <iostream>
int main() {
  const int size = 5;
  int arr[] = {5, 3, 8, 4, 6};
  int i, j;
  bool swapped;
  for (i = 0; i < size - 1; i++) {
    swapped = false;
    for (j = 0; j < size - 1; j++) {
      if (arr[j] > arr[j+1]) {
        int temp = arr[j];
        arr[j] = arr[j+1];
        arr[j+1] = temp;
        swapped = true;
      }
    }
    // IF no two elements were swapped by inner loop, then break
    if (swapped == false)
      break;
  }
  for (int k = 0; k < size; k++) {
    std::cout << arr[k]<< " ";
  }
}
```

Հաշվի չենք առնում որ վերջին էլեմենտները արդեն սորտավորված են

https://repl.it/@VahagVardanyan/BubbleFor

Ալգորիթմի իրականացում 2

```
#include <iostream>
int main() {
  const int size = 5;
  int arr[] = {5, 3, 8, 4, 6};
  int i, j;
  bool swapped;
  for (i = 0; i < size -1; i++) {
    swapped = false;
    for (j = 0; j < size - 1; j++) {
      if (arr[j] > arr[j+1]) {
        int temp = arr[j];
        arr[j] = arr[j+1];
        arr[j+1] = temp;
        swapped = true;
      }
    }
    // IF no two elements were swapped by inner loop, th
    if (swapped == false)
      break;
  }
  for (int k = 0; k < size; k++) {
    std::cout << arr[k]<< " ";
  }
}
```

https://repl.it/@VahagVardanyan/BubbleFor

Հաշվի չենք առնում որ վերջին էլեմենտները արդեն սորտավորված են

Այստեղ կարելի է տպել զանգվածը, քայլերի հերթականությունը ավելի լավ հասկանալու համար

Ալգորիթմի իրականացում 3

```
#include <iostream>
int main() {
  const int size = 5;
  int arr[] = {5, 3, 8, 4, 6};
  int i, j;
  bool swapped;
  for (i = 0; i < size -1 ; i++) {
    swapped = false;
    for (j = 0; j < size - i - 1; j++) {
      if (arr[j] > arr[j+1]) {
        int temp = arr[j];
        arr[j] = arr[j+1];
        arr[j+1] = temp;
        swapped = true;
      }
    }
    // IF no two elements were swapped by inner loop, then break
    if (swapped == false)
      break;
  }
  for (int k = 0; k < size; k++) {
    std::cout << arr[k]<< " ";
  }
}
```

Վերջին i էլեմենտները արդեն ճիշտ դասավորված են

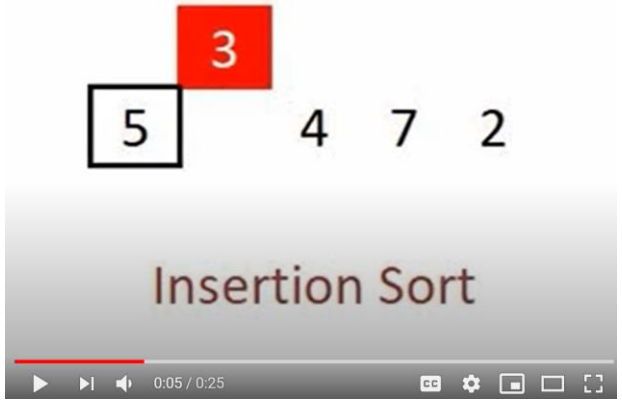
https://repl.it/@VahagVardanyan/BubbleFor

Ալգորիթմ 2: Ներդրմամբ սորտավորում

Մուտք՝ n չափանի ստատիկ «a» զանգված

Ելք՝ Զանգված սորտավորված աճման կարգով

1. Դիտարկել զանգվածի հերթական i էլեմենտը (i = 1 ից մինչև n-1)
2. Տեղափոխել a[i] էլեմենտը այնպես, որ այն հայտնվի իրենից ձախ ենթահաջորդականության մեջ ճիշտ սորտավորված դիրքում



https://www.youtube.com/watch?v=mTNC0ERo-ZI

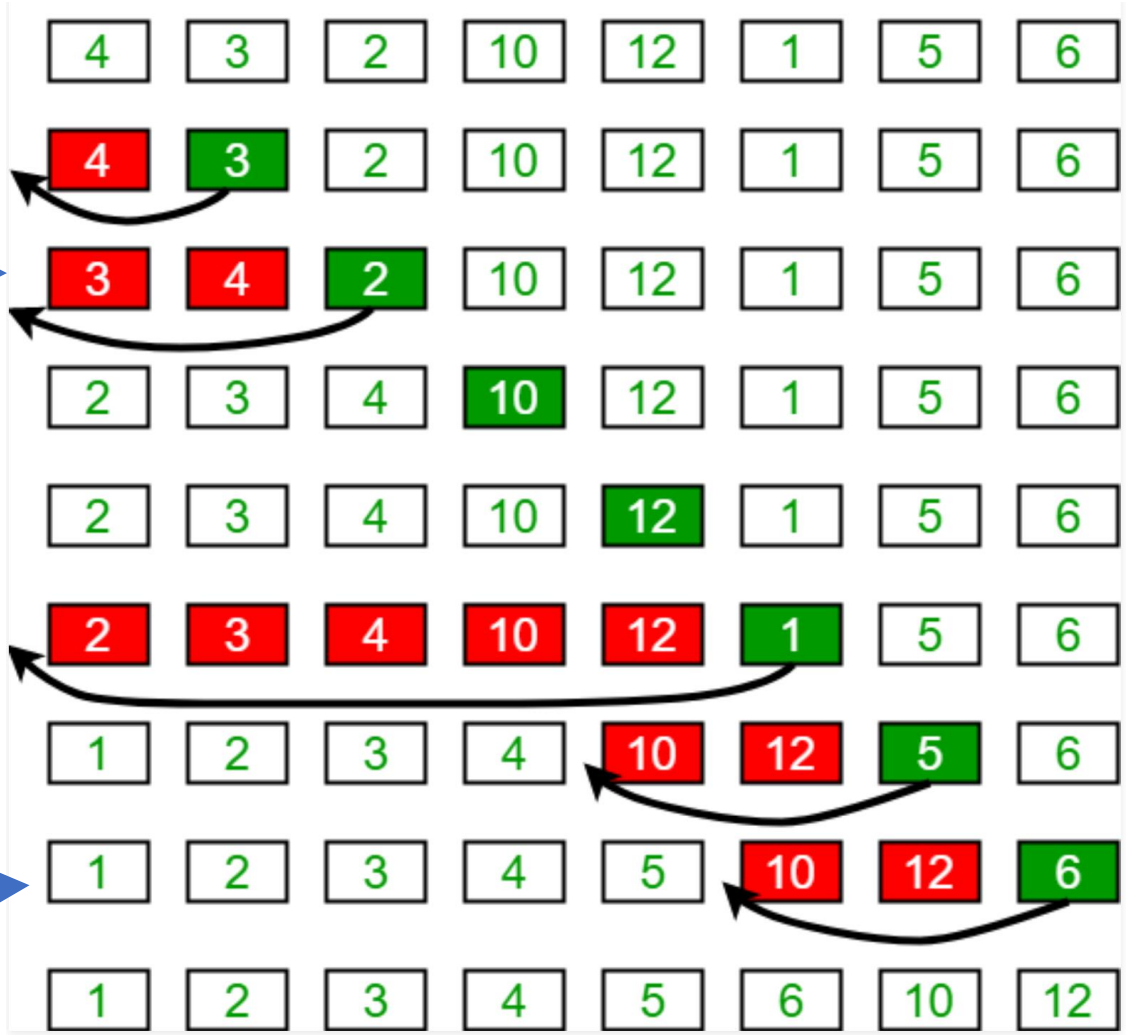
Այս սորտավորումը նման է թղթախաղում քարտերը դասավորելու պրոցեսին:

Ալգորիթմ 2: Ներդրմամբ սորտավորում

i=1, a[1]=3, պետք է տեղադրել a[0], a[1]-ի մեջ որպեսզի սորտավորված մնա ձախ մասը

i=2, a[2]=2, պետք է տեղադրել a[0], a[1], a[2]-ի մեջ որպեսզի սորտավորված մնա ձախ մասը

i=7, a[7]=6, պետք է տեղադրել a[0], a[1], a[2], … a[7]-ի մեջ որպեսզի սորտավորված մնա ձախ մասը



Ալգորիթմի իրականացում

```
#include <iostream>
int main() {
   const int size = 8;
   int arr[] = {14, 33, 27, 10, 35, 19, 42, 44};

   int i, key, j;
   for (i = 1; i < size; i++) {
       key = arr[i];
       j = i - 1;

       /* Move elements of arr[0..i-1], that are
       greater than key, to one position ahead
       of their current position */
       while (j >= 0 && arr[j] > key) {
          arr[j + 1] = arr[j];
          j = j - 1;
       }
       arr[j + 1] = key;
    }
    for (int k = 0; k < size; k++) {
       std::cout << arr[k]<< " ";
    }
}
```

https://repl.it/@VahagVardanyan/InsertionSort

Ալգորիթմի իրականացում

```
#include <iostream>
int main() {
   const int size = 8;
   int arr[] = {14, 33, 27, 10, 35, 19, 42, 44};

   int i, key, j;
   for (i = 1; i < size; i++) {
       key = arr[i];
       j = i - 1;

       /* Move elements of arr[0..i-1], that are
       greater than key, to one position ahead
       of their current position */
       while (j >= 0 && arr[j] > key) {
          arr[j + 1] = arr[j];
          j = j - 1;
       }
       arr[j + 1] = key;
    }
    for (int k = 0; k < size; k++) {
       std::cout << arr[k]<< " ";
    }
}
```

https://repl.it/@VahagVardanyan/InsertionSort

Այստեղ կարելի է տալ զանգվածը, քայլերի հերթականությունը ավելի լավ հասկանալու համար

Խնդիրներ

1. Գտնել զանգվածում կրկնվող էլեմենտների քանակը (սորտավորման կիրառմամբ)
2. Գտնել զանգվածում ամենաշատ կրկնվող էլեմենտը (սորտավորման կիրառմամբ)
3. Միաձուլել երկու սորտավորված զանգված իրար այնպես, որ որդյունքը լինի սորտավորված

Ալգորիթմների բարդության գնահատում

1. Ալգորիթմը կախված չէ ծրագրավորման լեզվից և համակարգչից:
2. Անհրաժեշտություն կա համեմատելու ալգորիթմների էֆեկտիվություն առանց հաշվի առնելու թե ինչ լեզվով է այն իրականացված կամ թե ինչ համակարգչի վրա այն աշխատաում
3. Այդ նպատակով օգտագործվում է ալգորիթմի բարդության ասիմտոտիկ գնահատական

Ալգորիթմների բարդության գնահատում

- Ալգորիթմի բարդության ասիմտոտիկ (մոտարկող) գնահատականը հիմնվում է հիպոթեթիկ համակարգչի վրա, որը աշխատում է հետևյալ կերպ.
  1. Բոլոր «պարզ» գործողությունները (+, *, -, /, =, if) կատարելու համար անհրաժեշտ է մեկ միավոր ժամանակ
  2. Ցիկլերը բաղկացած են մեկ և ավել «պարզ» հրամաններից: Ցիկլի կատարման ժամանակը կախված է ցիկլի իտերացիաների քանակից
- Ալգորիթմի կատարման ժամանակը հավասար է բոլոր գործողությունների կատարման ժամանակների գումարին:
- Ալգորիթմի բարդությունը գնահատվում է բոլոր հնարավոր մուտքային տվյալների համար
- Ալգորիթմի բարդությունը վատագույն դեպքում դա ալգորիթմի կատարած մաքսիմալ քայլերի քանակն է

Ալգորիթմների բարդության գնահատում

Մեծ O գնահատական:

O(n) – ալգորիթմի կատարած քայլերի քանակը **մոտավորապես** համարժեք է մուտքային տվյալների քանակին (գոյություն ունի C կոնստանտ թիվ, այնպես որ քայլերի քանակը <= c*n)

$O(n^2)$ - ալգորիթմի կատարած քայլերի քանակը **մոտավորապես** համարժեք է մուտքային տվյալների քանակի քառակուսուն (գոյություն ունի C կոնստանտ թիվ, այնպես որ քայլերի քանակը <= $c*n^2$)

O(1) - ալգորիթմի կատարած քայլերի քանակը կախված չէ մուտքային տվյալների քանակից (քայլերի քանակը կարող է լինել ցանկացած թիվ, բայց այդ թիվը կախված չէ մուտքային տվյալներից)

Ալգորիթմների բարդության գնահատում

- Ալգորիթմ: Տպել զանգվածի 80 - րդ էլեմենտը:
- Ալգորիթմի բարդությունը կախված չէ զանգվածի չափից:
- Ասիմտոտիկ գնահատական – O(1)

```
#include <iostream>

int main() {
  const int n = 100000;
  int a[n] = {};
  std::cout << a[80] << std::endl;

}
```

Ալգորիթմների բարդության գնահատում

- Ալգորիթմ: Գտնել զանգվածի ամենամեծ էլեմենտը:

```
#include <iostream>

1. int main() {
2.  const int n = 5;
3.  int arr[n];
4.  for (int i = 0; i < n; i++) {
5.     std::cin >> arr[i];
6.  }

7.  int max = arr[0];
8.  for (int i = 1; i < n; i++) {
9.     if (arr[i] > max) {
10.       max = arr[i];
11.     }
12.  }
13.  std::cout << "Max of array " << max << std::endl;
14. }
```

https://repl.it/@VahagVardanyan/ArrayMax

Ալգորիթմների բարդության գնահատում

- Ալգորիթմ: Գտնել զանգվածի ամենամեծ էլեմենտը:

n անգամ կատարվում է cin գործողությունը

```
#include <iostream>

1. int main() {
2.  const int n = 5;
3.  int arr[n];
4.  for (int i = 0; i < n; i++) {
5.     std::cin >> arr[i];
6.  }

7.  int max = arr[0];
8.  for (int i = 1; i < n; i++) {
9.     if (arr[i] > max) {
10.       max = arr[i];
11.     }
12.  }
13.  std::cout << "Max of array " << max << std::endl;
14. }
```

https://repl.it/@VahagVardanyan/ArrayMax

Ալգորիթմների բարդության գնահատում

- Ալգորիթմ: Գտնել զանգվածի ամենամեծ էլեմենտը:

```
#include <iostream>

1. int main() {
2.   const int n = 5;
3.   int arr[n];
4.   for (int i = 0; i < n; i++) {
5.      std::cin >> arr[i];
6.   }

7.   int max = arr[0];
8.   for (int i = 1; i < n; i++) {
9.      if (arr[i] > max) {
10.        max = arr[i];
11.      }
12.   }
13.   std::cout << "Max of array " << max << std::endl;
14. }
```

https://repl.it/@VahagVardanyan/ArrayMax

n անգամ կատարվում է cin գործողությունը

n - 1 անգամ կատարվում է > գործողությունը
n – 1 անգամ կատարվում է = գործողությունը

Ալգորիթմների բարդության գնահատում

- Ալգորիթմ: Գտնել զանգվածի ամենամեծ էլեմենտը:

```
#include <iostream>

1. int main() {
2.   const int n = 5;
3.   int arr[n];
4.   for (int i = 0; i < n; i++) {
5.      std::cin >> arr[i];
6.   }

7.   int max = arr[0];
8.   for (int i = 1; i < n; i++) {
9.      if (arr[i] > max) {
10.        max = arr[i];
11.      }
12.   }
13.   std::cout << "Max of array " << max << std::endl;
14. }
```

https://repl.it/@VahagVardanyan/ArrayMax

n անգամ կատարվում է cin գործողությունը

n - 1 անգամ կատարվում է > գործողությունը
n – 1 անգամ կատարվում է = գործողությունը

Ընդհանուր գործողություններ գումարը հավասար է 3n -2:
Ասիմտոտիկ գնահատական - O(n).

Պղպջակ ալգորիթմի բարդության գնահատում

```
#include <iostream>
int main() {
  const int n = 5;
  int arr[n] = {5, 3, 8, 4, 6};
  int i, j;
  bool swapped;
  for (i = 0; i < n; i++) {
    swapped = false;
    for (j = 0; j < n - i - 1; j++) {
      if (arr[j] > arr[j+1]) {
        int temp = arr[j];
        arr[j] = arr[j+1];
        arr[j+1] = temp;
        swapped = true;
        }
    }
   // IF no two elements were swapped by inner loop, then
   if (swapped == false)
     break;
   }
   for (int k = 0; k < n; k++) {
     std::cout << arr[k]<< " ";
   }
}
```

https://repl.it/@VahagVardanyan/BubbleFor

Ամեն i-երրորդ էլեմենտի համար կատարվում է c* (n-i-1) գործողություն, որտեղ c - ն ցիկլի պարզ գործուղությունների քանակն է

Պղպջակ ալգորիթի բարդության գնահատում

```
#include <iostream>
int main() {
  const int n = 5;
  int arr[n] = {5, 3, 8, 4, 6};
  int i, j;
  bool swapped;
  for (i = 0; i < n; i++) {
    swapped = false;
    for (j = 0; j < n - i - 1; j++) {
      if (arr[j] > arr[j+1]) {
        int temp = arr[j];
        arr[j] = arr[j+1];
        arr[j+1] = temp;
        swapped = true;
        }
    }
   // IF no two elements were swapped by inner loop, the
   if (swapped == false)
     break;
   }
   for (int k = 0; k < n; k++) {
     std::cout << arr[k]<< " ";
   }
}
```

https://repl.it/@VahagVardanyan/BubbleFor

c* (n-i-1) գործողությունները կրկնվում են n անգամ:

Ամեն i-երրորդ էլեմենտի համար կատարվում է c* (n-i-1) գործողություն, որտեղ c - ն ցիկլի պարզ գործուղությունների քանակն է

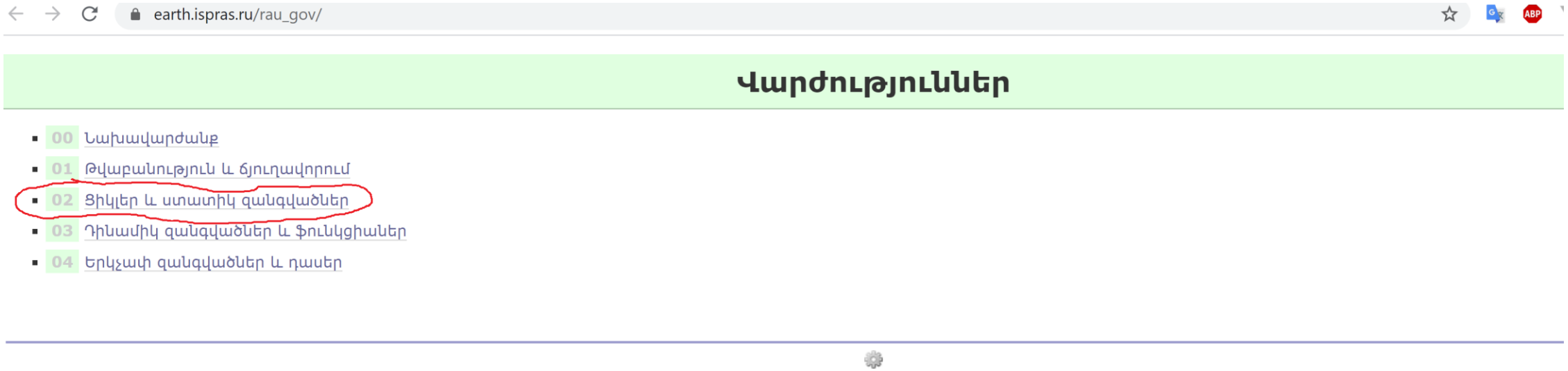
Պղպջակ ալգորիթի բարդության գնահատում

```
#include <iostream>
int main() {
  const int n = 5;
  int arr[n] = {5, 3, 8, 4, 6};
  int i, j;
  bool swapped;
  for (i = 0; i < n; i++) {
    swapped = false;
    for (j = 0; j < n - i - 1; j++) {
      if (arr[j] > arr[j+1]) {
        int temp = arr[j];
        arr[j] = arr[j+1];
        arr[j+1] = temp;
        swapped = true;
        }
    }
   // IF no two elements were swapped by inner loop, then break
   if (swapped == false)
     break;
   }
   for (int k = 0; k < n; k++) {
     std::cout << arr[k]<< " ";
   }
}
```

https://repl.it/@VahagVardanyan/BubbleFor

c* (n-i-1) գործողությունները կրկնվում են n անգամ.
i=0 կկատարի **c * (n – 1)**
i=1 կկատարի **c * (n – 2)**
……
i=n-2 կկատարի **c**
Ընդհանուր գումարը կստացվի **c * (n - 1) * (n - 2) / 2** (թվաբանական պրոգրեսիա)։
Այսպիսով այս ալգորիթմի բարդությունը $O(n^2)$

Տնային աշխատանք



https://earth.ispras.ru/rau_gov/

Տնային աշխատանք

- Խնդիրներ 18 - 20

Շնորհակալություն. Հարցե՞ր