ZeceLaInfo

Recursivitate

Paul-Ioan Somesan

ZeceLaInfo

Date generale

- Recursivitatea este proprietatea unor notiuni de a se defini prin ele insele. (PbInfo.ro)
- Exemple de recursivitati cunoscute de la matematica (recurenta):
  - Sirul lui Fibonacci
  - Progresiile matematice (aritmetice si geometrice)
  - Calculul Factorialelor
- La informatica, rolul recursivitatii este de a tine locul structurilor repetitive si de a estetiza si structura mai bine codul. Deoarece sunt similare cu structurile repetitive, vom avea nevoie de o initializare (apel), de o conditie de oprire si de o modificare spre conditia de oprire (auto-apel).

ZeceLaInfo

Functii Recursive Matematice

```
6   int factorial(int n){
7       if(n <= 1)
8           return 1;
9       else return n * factorial(n - 1);
10  }
```

```
6   int fibonacci(int n){
7       if(n == 1 || n == 2)
8           return 1;
9       return fibonacci(n-1) + fibonacci(n-2);
10  }
```

```
6   int nr_div(int n, int d){
7       if(n % d == 0 && d * d < n)
8           return 2 + nr_div(n, d+1);
9       if(d * d == n)
10          return 1;
11      if(d < n)
12          return nr_div(n, d + 1);
13      return 0;
14  }
```

```
6   int sum_div(int n, int d){
7       if(n % d == 0 && d * d < n)
8           return n / d + d + sum_div(n, d+1);
9       if(d * d == n)
10          return d;
11      if(d < n)
12          return sum_div(n, d + 1);
13      return 0;
14  }
```

ZeceLaInfo

Functii Recursive II

```
6   int sum_cif(int n){
7       if(n == 0)
8           return 0;
9       return n % 10 + sum_cif(n / 10);
10  }
```

```
6   void baza2(int n){
7       if(n == 0)
8           return ;
9       baza2(n / 2);
10      cout << n % 2;
11  }
```

```
4   int numar_cifre(int n){
5       if(n < 10)
6           return 1;
7       return 1 + numar_cifre(n / 10);
8   }
```

```
6   int sum_vector(int a[], int st, int dr){
7       if(st > dr)
8           return 0;
9       return a[st] + sum_vector(a, st+1, dr);
10  }
```

ZeceLaInfo

Frumusetea functiilor Recursive

- Motivul pentru care folosim foarte mult functiile Recursive este pentru ca acestea ne ajuta sa construim solutii mai estetice, mai apropiate de rationamentul uman si mult mai intuitive.
  - Spre exemplu, implementare sirurile recurente de la matematica este mult mai usoara folosind functii recursive decat folosind functii iterative.
- Un alt motiv pentru care folosim functiile Recursive este pentru ca putem inversa pasii (vezi exemplul cu scrierea in baza 2). In functie de pozitionarea fata de auto-apelul functiei, vom face operatiile dorite la momentul auto-apelului sau la intoarcerea din adancime.
- Functiile BackTracking sunt foarte usor de implementat recursiv.

ZeceLaInfo

Aspectele neplacute ale recursivitatii

- TOATE apelurile recursive sunt stocate pe stiva programului, lucru care duce la riscul de a depasi cu mult memoria alocata programului.
- Functiile recursive sunt foarte usor de scapat de sub control in lipsa unui algoritm extrem de bun si eficient.
- Pot fi extrem de imprevizibile. (Complexitatea unei functii recursive poate fi foarte greu de estimat)
- Pot fi dificil de simulat, motiv pentru care grilele date la Examenul de Admitere la UBB care contin functii recursive vor fi o reala provocare pentru toata lumea.

ZeceLaInfo

Divide et Impera

- Divide et Impera este un concept filozofic antic care are in vedere un principiu foarte simplu: pentru a controla, trebuie sa divizi.
- La informatica, acest lucru se foloseste usor diferit. Folosim Divide et Impera, 90% dintre cazuri, pentru a determina stari si caracteristici ale structurilor de date.
- Ideea din spate este foarte simpla: daca dorim determinarea starii unui vector, este imposibil sa stim, uitandu-ne la el, asa ca incepem injumatatirea in mod repetitiv a vectorului pana cand ajungem la un singur element. Recursiv, la intoarcere, construim rezultatul asteptat.

Exemple de Divide et Impera

```
6   int sum_vector(int a[], int st, int dr){
7       if(st == dr)
8           return a[st];
9       int mij = (st + dr) / 2;
10      return sum_vector(a, st, mij) + sum_vector(a, mij + 1, dr);
11  }
```

```
6   bool toate_pare(int a[], int st, int dr){
7       if(st == dr)
8           return a[st] % 2 == 0;
9       int mij = (st + dr) / 2;
10      return toate_pare(a, st, mij) && toate_pare(a, mij + 1, dr);
11  }
```

Preferatul meu:

Cerința

Se dă un vector cu `n` elemente numere naturale. Folosind metoda **Divide et Impera** să se verifice dacă are elementele ordonate crescător.

```
4   bool verif_ord(int a[], int st, int dr){
5       if(st == dr)
6           return true;
7       int mij = (st + dr) / 2;
8       return verif_ord(a, st, mij) && verif_ord(a, mij + 1, dr) && a[mij] <= a[mij + 1];
9   }
```

ZeceLaInfo

Cautare Binara cu Divide et Impera

```
6   bool cb(int a[], int st, int dr, int val){
7       if(st == dr)
8           return a[st] == val;
9       int mij = (st + dr) / 2;
10      if(a[mij] == val)
11          return true;
12      if(a[mij] < val)
13          return cb(a, mij + 1, dr, val);
14      return cb(a, st, mij - 1, val);
15  }
```

ZeceLaInfo

Metode de sortare

- In spatele celor mai rapide metode de Sortare din algoritmica sta conceptul de Divide et Impera.
- Pentru Examenul de Admitere la UBB 2023, va trebui sa stapaniti foarte bine 2 metode de sortare (apar pe programa de examen):
  - Sortare MergeSort – Complexitate O(n * log2(n))
  - Sortare QuickSort – Complexitate O(n * log2(n)) – caz mediu
- Pe langa acestea, dintre metodele de Sortare invatate in clasa a IX-a, va trebui sa stapaniti la cel mai inalt nivel si:
  - Selection Sort
  - Insertion Sort
  - Bubble Sort
  - Counting Sort

ZeceLaInfo

Quick Sort

```cpp
#include <iostream>
using namespace std;

int a[100001], n;

int partitie(int st, int dr){
    int pivot = a[dr];
    int index_curent = st;
    for(int i = st; i < dr; ++i){
        if(a[i] <= pivot){
            swap(a[i], a[index_curent]);
            index_curent++;
        }
    }
    swap(a[dr], a[index_curent]);
    return index_curent;
}

```

```cpp
void QuickSort(int st, int dr){
    if(st <= dr){
        int pindex = partitie(st, dr);
        QuickSort(st, pindex - 1);
        QuickSort(pindex + 1, dr);
    }
}

int main(){
    cin >> n;
    for(int i = 1; i <= n; ++i)
        cin >> a[i];
    QuickSort(1, n);
    for(int i = 1; i <= n; ++i)
        cout << a[i] << ' ';
    return 0;
}
```

ZeceLaInfo

Merge Sort

```
6   void mergeSort(int st, int dr){
7       if(st == dr)
8           return ;
9       else{
10          int mij = (st + dr) / 2;
11          mergeSort(st, mij);
12          mergeSort(mij + 1, dr);
13          int inda = st, indb = mij + 1, indc = 0;
14          while(inda <= mij && indb <= dr)
15              if(a[inda] <= a[indb])
16                  c[++indc] = a[inda++];
17              else c[++indc] = a[indb++];
18          while(inda <= mij)
19              c[++indc] = a[inda++];
20          while(indb <= dr)
21              c[++indc] = a[indb++];
22          for(int i = 1; i <= indc; ++i)
23              a[st + i - 1] = c[i];
24      }
25  }
```

ZeceLaInfo

Va multumesc mult pentru atentie!

- In cazul in care aveti intrebari / doriti sa vorbiti cu mine despre Admitere, Informatica sau pur si simplu vreti sa ma contactati, astept cu tot dragul mesajele voastre pe una dintre caile:


@paul.somesan


@zece_la_info


paul.somesan02@gmail.com

academia-zecelainfo.com