ساختمان داده‌ها و الگوریتم‌ها

فصل سوم

صمد نجارقابل

S.Najjar.G@Gmail.com

S.Najjar.G@Gmail.com

در جلسات قبل ...

- مقدمه‌ای بر ساختمان داده
- تعاریف اولیه
- مفاهیم پایه درباره پیچیدگی زمانی
- معرفی پیچیدگی زمانی و حافظه‌ای
- نحوه پیدا کردن گام های برنامه
- نمادهای خاص پیچیدگی زمانی
- نحوه محاسبه انواع پیچیدگی ها و روش‌های حل آن

S.Najjar.G@Gmail.com

فهرست مطالب

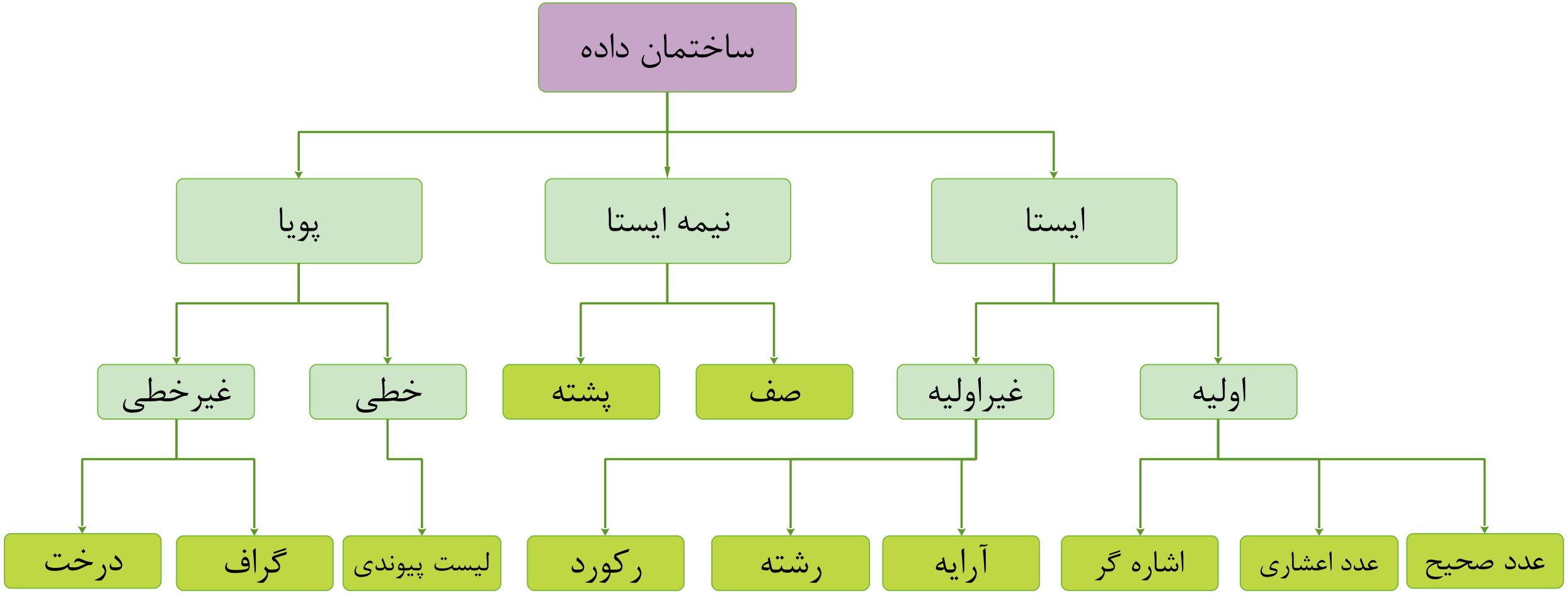
S.Najjar.G@Gmail.com

تعاریف اولیه ...

- **داده ایستا:** از فضای محدود و از پیش‌تعریف شده استفاده کرده و این فضا در طول برنامه ثابت است.
- **داده پویا:** با استفاده از اشاره گرها امکان تغییر نامحدود و پویا متناسب با داده‌ها را فراهم می‌کند.
- **داده‌های نیمه ایستا:** برخی از ساختمان داده ها را هم به صورت محدود با استفاده از آرایه (ساختار ایستا) و هم به صورت نامحدود با استفاده از لیست پیوندی (ساختار پویا) پیاده‌سازی کرد

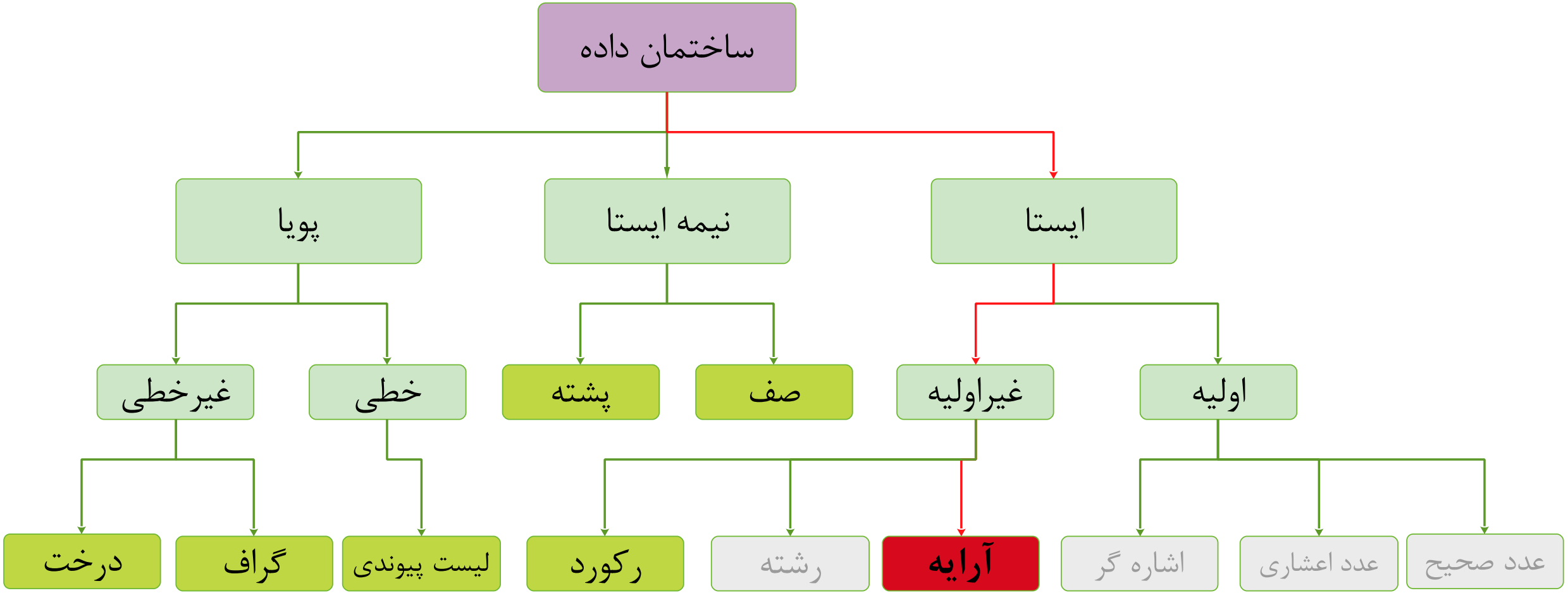
S.Najjar.G@Gmail.com

دسته بندي ساختمان داده‌ها



S.Najjar.G@Gmail.com

دسته بندي ساختمان داده‌ها



S.Najjar.G@Gmail.com

سوال

- آرایه چه نوع ساختاری است؟
- چرا از آرایه استفاده می‌کنیم؟
- نقاط قوت و ضعف این نوع ساختار داده چیست؟

S.Najjar.G@Gmail.com

Introduction Array

نوع داده مجرد یا انتزاعی (ADT): نوع داده‌ای است که در آن مشخصات داده‌ها و اعمال بر روی آنها از بازنمایی و پیاده‌سازی داده جدا می‌شود.

Arrays

S.Najjar.G@Gmail.com

Array:

a set of index and value

data structure:

For each index, there is a value associated with that index.

Representation:

implemented by using consecutive memory.

Array (Cont.)

S.Najjar.G@Gmail.com

Structure *Array* is

objects: A set of pairs <*index*, *value*> where for each value of *index* there is a value from the set *item*. *Index* is a finite ordered set of one or more dimensions, for example, {0, … , n-1} for one dimension, {(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2)} for two dimensions, etc.

Functions:

for all A ∈ Array, *i* ∈ *index*, x ∈ *item*, *j*, *size* ∈ integer

Array Create(j, list) ::= **return** an array of *j* dimensions where list is a j-tuple whose *i*th element is the size of the *i*th dimension. *Items* are undefined.

Item Retrieve(A, *i*) ::= **if** (*i* ∈ *index*) **return** the item associated with index value *i* in array A **else return** error

Array Store(A, *i*, *x*) ::= **if** (*i* in *index*) **return** an array that is identical to array A except the new pair <*i*, *x*> has been inserted **else return** error

end array

Array (Cont.)

S.Najjar.G@Gmail.com

```
int list[5], *plist[5];
```

list[5]: five integers
list[0], list[1], list[2], list[3], list[4]
*plist[5]: five pointers to integers
plist[0], plist[1], plist[2], plist[3], plist[4]

Implementation of 1-D array:

list[0]	base address = α
list[1]	α + sizeof(int)
list[2]	α + 2*sizeof(int)
list[3]	α + 3*sizeof(int)
list[4]	α + 4*size(int)

Array (Cont.)

S.Najjar.G@Gmail.com

Compare int *list1 and int list2[5] in C.

Same: list1 and list2 are pointers.
Difference: list2 reserves five locations.

Notations:

list2 is a pointer to list2[0]
(list2 + i) is a pointer to list2[i] (&list2[i])
*(list2 + i) is list2[i]

Array (Cont.)

S.Najjar.G@Gmail.com

Example: 1-dimension array addressing

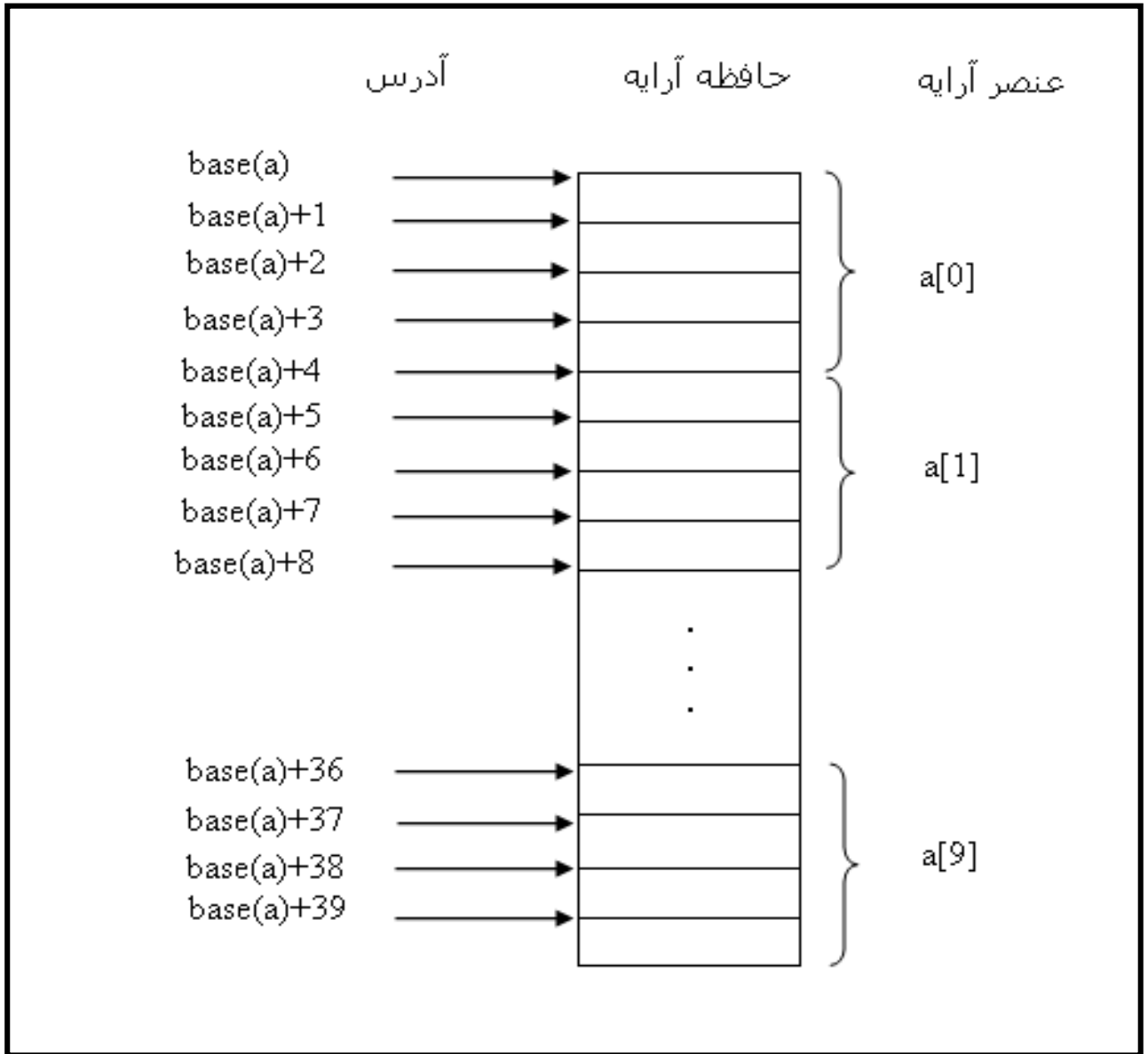
int one[] = {0, 1, 2, 3, 4};
Goal: print out address and value

```
void print1(int *ptr, int rows)
{
/* print out a one-dimensional array using a pointer */
        int i;
        printf("Address Contents\n");
        for (i=0; i < rows; i++)
                printf("%8u%5d\n", ptr+i, *(ptr+i));
        printf("\n");
}
```

1-dimension array

S.Najjar.G@Gmail.com

Float a[10];



Array (Cont.)

S.Najjar.G@Gmail.com

مشخص کردن حافظه برای عناصر آرایه یک بعدی:

A: Array[L…U] of type

اگر type نوع عناصر آرایه به اندازه n باشد و اگر فرض کنیم آدرس شروع آرایه α باشد آنگاه:

- تعداد عناصر آرایه A ==== $U - L + 1$
- حافظه ی تخصیصی به این آرایه ===== $(U - L + 1) \times n$
- آدرس شروع عنصر A[i] ==== $(i - L) \times n + \alpha$

S.Najjar.G@Gmail.com

Array example

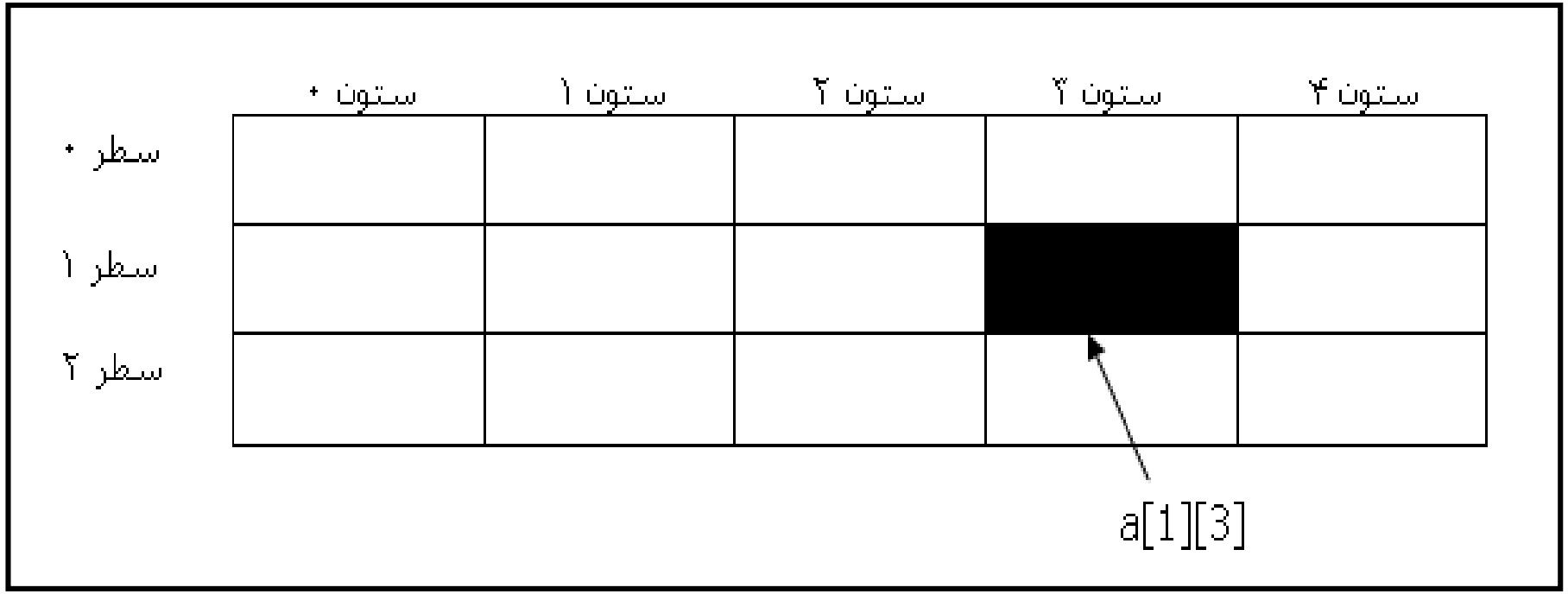
آرایه‌ی Test در زبان C به صورت زیر تعریف شده است. اگر آدرس شروع این آرایه ۲۵۰۰ باشد. مطلوب است آدرس عنصر Test[30] را بیابید.

```
float Test[200];
```

$$\text{Address (Test[30]) = \& Test[30]= } (30 - 0) \times 4 + 2500 = 2620$$

2-dimension array

S.Najjar.G@Gmail.com



2-dimension array

S.Najjar.G@Gmail.com

مشخص کردن حافظه برای عناصر آرایه دو بعدی:

A: $Array[L_1 \dots U_1, L_2 \dots U_2]\ of\ type$

اگر type نوع عناصر آرایه (که هر عنصر از این نوع به اندازه n است) باشد و اگر فرض کنیم آدرس شروع این آرایه α باشد آنگاه:

- تعداد عناصر آرایه A ==== $(U_1 - L_1 + 1) \times (U_2 - L_2 + 1)$
- حافظه ی تخصیصی به این آرایه === $(U_1 - L_1 + 1) \times (U_2 - L_2 + 1) \times n$
- آدرس شروع عنصر A[i,j] در حالتی که آرایه سطری ذخیره شده است = $[(i - L_1) \times (U_2 - L_2 + 1) + (j - L_2)] \times n + \alpha$
- آدرس شروع عنصر A[i,j] در حالتی که آرایه ستونی ذخیره شده است = $[(j - L_2) \times (U_1 - L_1 + 1) + (i - L_1)] \times n + \alpha$

S.Najjar.G@Gmail.com

ماتریس اسپارس

- در ریاضیات ، یک ماتریس شامل m سطر و n ستون از اعضا می باشد
- ماتریسی که عناصر صفر آن زیاد بوده ماتریس اسپارس نامیده می شود.

	col 0	col 1	col 2
row 0	−27	3	4
row 1	6	82	−2
row 2	109	−64	11
row 3	12	8	9
row 4	48	27	47

5*3

(a) 15/15

	col 0	col 1	col 2	col 3	col 4	col 5
row 0	15	0	0	22	0	−15
row 1	0	11	3	0	0	0
row 2	0	0	0	−6	0	0
row 3	0	0	0	0	0	0
row 4	91	0	0	0	0	0
row 5	0	0	28	0	0	0

6*6

(b) 8/36

sparse matrix data structure?

S.Najjar.G@Gmail.com

ماتریس اسپارس

می توان از یک آرایه از سه تایی های <row, col, value> برای نمایش یک ماتریس اسپارس استفاده کرد.

	col 0	col 1	col 2	col 3	col 4	col 5
row 0	15	0	0	22	0	−15
row 1	0	11	3	0	0	0
row 2	0	0	0	−6	0	0
row 3	0	0	0	0	0	0
row 4	91	0	0	0	0	0
row 5	0	0	28	0	0	0

	row	col	value
a[0]	6	6	8
[1]	0	0	15
[2]	0	3	22
[3]	0	5	−15
[4]	1	1	11
[5]	1	2	3
[6]	2	3	−6
[7]	4	0	91
[8]	5	2	28

(a)

S.Najjar.G@Gmail.com

ماتریس اسپارس

می توان از یک آرایه از سه تایی های <row, col, value> برای نمایش یک ماتریس اسپارس استفاده کرد.

	col 0	col 1	col 2	col 3	col 4	col 5
row 0	15	0	0	22	0	−15
row 1	0	11	3	0	0	0
row 2	0	0	0	−6	0	0
row 3	0	0	0	0	0	0
row 4	91	0	0	0	0	0
row 5	0	0	28	0	0	0

# of rows (columns)

	row	col	value
a[0]	6	6	8
[1]	0	0	15
[2]	0	3	22
[3]	0	5	−15
[4]	1	1	11
[5]	1	2	3
[6]	2	3	−6
[7]	4	0	91
[8]	5	2	28

(a)

S.Najjar.G@Gmail.com

ماتریس اسپارس

می توان از یک آرایه از سه تایی های <row, col, value> برای نمایش یک ماتریس اسپارس استفاده کرد.

	col 0	col 1	col 2	col 3	col 4	col 5
row 0	15	0	0	22	0	−15
row 1	0	11	3	0	0	0
row 2	0	0	0	−6	0	0
row 3	0	0	0	0	0	0
row 4	91	0	0	0	0	0
row 5	0	0	28	0	0	0

# of nonzero terms

# of rows (columns)

	row	col	value
a[0]	6	6	8
[1]	0	0	15
[2]	0	3	22
[3]	0	5	−15
[4]	1	1	11
[5]	1	2	3
[6]	2	3	−6
[7]	4	0	91
[8]	5	2	28

(a)

S.Najjar.G@Gmail.com

ماتریس اسپارس

می توان از یک آرایه از سه تایی های <row, col, value> برای نمایش یک ماتریس اسپارس استفاده کرد.

row, column in ascending order

# of nonzero terms

# of rows (columns)

	col 0	col 1	col 2	col 3	col 4	col 5
row 0	15	0	0	22	0	−15
row 1	0	11	3	0	0	0
row 2	0	0	0	−6	0	0
row 3	0	0	0	0	0	0
row 4	91	0	0	0	0	0
row 5	0	0	28	0	0	0

	row	col	value
a[0]	6	6	8
[1]	0	0	15
[2]	0	3	22
[3]	0	5	−15
[4]	1	1	11
[5]	1	2	3
[6]	2	3	−6
[7]	4	0	91
[8]	5	2	28

(a)

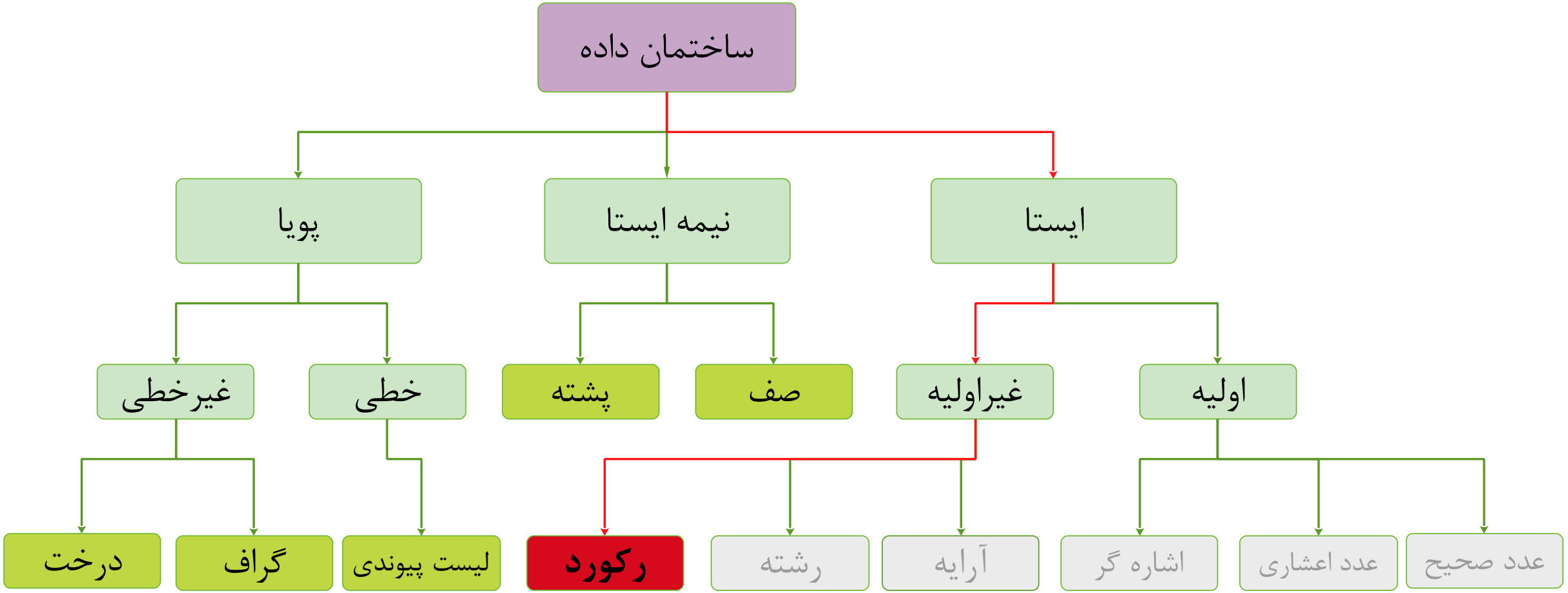
S.Najjar.G@Gmail.com

Home Work 4

- برنامه ای بنویسید که ماتریس A (n*n) را که یک ماتریس سپارس است بگیرد و به صورت که حافظه کمی استفاده نمایید ذخیره کنید. (حداکثر مقدار n ۵۰ می‌باشد، سعی شود برای ماتریس اولیه که از ورودی گرفته می‌شود حافظه تخصیص داده نشود)

S.Najjar.G@Gmail.com

دسته بندي ساختمان داده‌ها



Structures (records)

S.Najjar.G@Gmail.com

- **ساختار**: مجموعه‌ای از عناصر هست که لزومی ندارد داده‌های آن یکسان باشد.

```
struct {
        char name[10];
        int age;
        float salary;
        } person;

strcpy(person.name, “james”);
person.age=10;
person.salary=35000;
```