فایل اول درس ساختمان های داده

دانشجویان عزیز کلاس درس ساختمان های داده

- با آرزوی سلامتی برای تمامی شما، در فایل ارسالی سعی گردیده تمامی مطالب، جهت مطالعه دقیق و انجام تمرین با جزییات مطرح گردد.

موفق و سلامت باشید-اسبقی

مرور مفاهیم کلی و اهمیت مبحث داده برای رشته مهندسی کامپیوتر:

- **بررسی درس های پیشنیاز درس ساختمان های داده**

  - **مبانی کامپیوتر**
  - **برنامه نویسی پیشرفته**

- **مروری بر چندین الگوریتم برنامه های مورد بحث در کلاس**

- **برنامه ای بنویسید که 10 عدد صحیح را بعنوان ورودی گرفته و میانگین آنها را محاسبه کند؟**
- **برنامه ای بنویسید که 100 عدد اعشاری را بعنوان ورودی گرفته و ماکزیمم میان آنها را محاسبه کند؟**
- **برنامه ای بنویسید که طول و عرض مستطیلی را بعنوان ورودی گرفته و محیط و مساحت آنها را محاسبه کند؟**
- **برنامه ای بنویسید که فاکتوریل یک عدد صحیح را محاسبه کند؟**
- **برنامه ای بنویسید که 10 کاراکتر را بعنوان ورودی گرفته و آنها را مرتب نماید؟**

- **بر همین اساس داریم:**

برنامه = الگوریتم + داده

- **مفاهیمی که در ابتدای درس ساختمان های داده در کلاس، نیاز به بحث و همفکری دارند:**

- **نقش و اهمیت داده در دنیای امروز**
- **مفهوم الگوریتم**
- **مفهوم ساختمان های داده**

الگوریتم و معیارهای ارزیابی آن:

طبق موارد مطرح شده در کلاس، برای حل یک مسأله، راه حل های گوناگونی وجود دارد پس:

- برای حل یک مسأله برنامه نویسی، می توان الگوریتم های متنوعی را ارائه نمود.

بعنوان مثال، اگر یک مسأله برنامه نویسی در کلاس مطرح شود، هر دانشجو به روش خاص خود الگوریتمی را ارائه خواهد نمود، حال اگر نیاز به انتخاب الگوریتم بهینه باشد، معیار ارزیابی الگوریتم ها چیست؟

معیار ارزیابی الگوریتم ها:

- **پیچیدگی زمانی**
- **بهره وری حافظه**

سوال: در دنیای امروز، با توجه به قیمت و ظرفیت حافظه ها، اهمیت کدام معیار پررنگ تر است؟

پیچیدگی زمانی:

سوال: برای محاسبه پیچیدگی زمانی الگوریتم ها، آیا زمان های واقعی مدنظر می باشد؟

مشکلات محاسبه زمان واقعی اجرای الگوریتم ها:

- تفاوت مشخصات فنی سیستم های مختلف که برای تست استفاده می شوند.
- تفاوت در زمان پاسخگویی استفاده از یک سیستم در شرایط و زمان های مختلف.

محاسبه پیچیدگی زمانی الگوریتم ها با شمارش تعداد دستورالعمل های الگوریتم:

```
int  function(int   A[] ,int   n)
{ int  i, sum = 0;
    for(i = 0; i<n ; i++)
       sum+=A[i];
   cout<<"sum ="<<sum;
   return  sum;}
```

سوال: آیا وزن تمامی دستورات به یک شکل است؟ نحوه محاسبه و شمارش تعداد دستورات چگونه است؟

شمارش تعداد دستورالعمل های الگوریتم برای محاسبه پیچیدگی زمانی الگوریتم ها:

معیار شمارش، دستوراتی می باشند که بیشترین تعداد تکرار را در الگوریتم دارند.
پس برای محاسبه زمان و پیچیدگی زمانی الگوریتم ها، نیاز به شمارش دستور العمل های الگوریتم البته دستوراتی با بیشترین تعداد تکرار وجود دارد. بر همین اساس و در ادامه درس، تابع پیچیدگی زمانی برای ساختارهای تکرار در الگوریتم ها محاسبه می گردد.

ساختار های تکرار:

- حلقه for
- حلقه while/do while
- الگوریتم های بازگشتی

حلقه while /do while:

حلقه while را با ذکر مثال بررسی نموده تا نحوه محاسبه تابع پیچیدگی زمانی مشخص گردد:

تابعی طراحی نمایید که یک آرایه ای از اعداد صحیح و یک عدد صحیح را بعنوان ورودی گرفته و بررسی نماید که عدد مذکور در آرایه وجود دارد یا خیر؟

```
int  function(int   A[n] ,int   x)
{ int  i=0;
    while( i<n && A[i]≠ x)
       i++;
   if( i >= n)
     return  -1;
 else
    return   i;}
```

برای محاسبه تابع پیچیدگی زمانی در الگوریتم هایی که شامل حلقه while می باشند، نمیتوان مانند حلقه for یک راهکار ثابت را بکار برد زیرا در ساختار حلقه while ، تعداد تکرار حلقه کاملا متاثر از شرط حلقه بوده و برای آن لازم است دو حالت بهترین و بدترین بسته به شرط حلقه محاسبه گردد.

در این الگوریتم اگر بعنوان مثال آرایه A و عدد مورد جستجوی x بصورت حالت اول باشد:

A	50	30	10	20	70	60	100	90

بعنوان مثال اگر عدد x=50 باشد، حلقه while در اولین تکرار x را یافته و متوقف می گردد. پس فقط یک دور تکرار خواهد داشت.(بهترین حالت)

ولی بعنوان مثال اگر x=80 یا x=90 باشد، حلقه while بطور کامل اجرا شده و حداکثر دور تکرار را خواهد داشت.(بدترین حالت)

B(n) = 1 (Best case)

W(n) = n (worst case)

T(n) = (n+1)/2

البته در مورد حلقه while، همیشه محاسبه تابع پیچیدگی به این سادگی نبوده و نیاز به درنظرگرفتن قوانین احتمال وقوع هر کدام از حالتهای بهترین و بدترین وجود دارد.