دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

درس تحلیل و طراحی الگوریتم‌ها

راه‌حل تکالیف:

سری دوم – تقسیم و حل

نام استاد درس:

دکتر بهروز شاهقلی

نام دستیاران آموزشی درس:

رضا رستگاری

جواد جعفری

زهرا تاکی

سارا کهتری

امیرحسین عرب‌پور

میلاد محمدی

نیم‌سال دوم تحصیلی ۱۴۰۰-۱۴۰۱

فهرست

تمرین اول: باز سالو

الف)

1.

$$f(n) = 0.5 \ f(n-1)+1 \quad , f(1) = 1$$

با امتحان کردن چند مقدار اولیه میتوانیم رابطه ((f(n) = 2(1-1/(2^(n را حدس بزنیم.
حال باید حدسمان را با استقرای ضعیف اثبات کنیم.

پایه استقرا :

که با توجه به فرض برای پایه رابطه بازگشتی،

n=1 : f(1) = 2(1-(1/2) = 1

پایه اثبات می‌شود.

فرض استقرا :

n = k : (f(k) = 2(1-1/(2^(k)

حکم استقرا :

n = k+1 : f(k+1) = 2(1-1/(2^(k+1)

اثبات حکم :

f(k+1) = 1/2(f(k)+1

(طبق فرض استقرا) —>

½(2(1-1/(2^k))+1 = 2+1/(2^(k+1)) = 2(1+1/(2^(k+1)))

که همان حکم است و بنابرین حکم استقرا اثبات میشود.

2.

$$f(n) = 5 \ f(n-1) \ -6f(n-2)$$
$$f(1) = -1,$$
$$f(2)=1$$

معادله مشخصه مربوط به رابطه بازگشتی:

r^2 - 5r + 6 = 0 → (r-2)(r-3) = 0

و بنابرین ریشه های معادله بازگشتی برابر با r1 = 3 , r2 = 2 میباشند و فرم کلی جواب به شکل

f(n) = a3^n + b 2^n

خواهد بود.

حال با استفاده از مقادیر پایه n=1 , n=2 که از پیش داده شده اند، دستگاهی با دو معادله و دو مجهول تشکیل میدهیم و a,b را بدست می آوریم.

$$3a+2b = -1$$
$$9a + 4b = 1$$

که از حل دستگاه به دست می آید a=1 , b =-2 و در نتیجه فرم بسته f(n) برابر با

$$f(n) = 3^n - 2(n+1)$$

خواهد بود.

3.

$$f(n) = 4f(n-1)-3f(n-2)+2^n ,$$
$$f(1) = -1,$$
$$f(2)=11$$

رابطه بازگشتی داده شده خطی و نا همگن است. راه های مختلفی را میتوان برای حل این معادله به کار برد اما در اینجا از قضیه B.3 گفته شده در پیوست B کتاب نیوپولیتان استفاده میکنیم.

بخش ناهمگن رابطه برابر با2^n است که به ما بخش (r-2) را میدهد.
و از بقیه رابطه به همراه این بخش خواهیم داشت:

$$(r^2 - 4r + 3)(r-2) = (r-1)(r-3)(r-2) =0$$

که به ما سه ریشه متفاوت r = 1,2,3 را میدهد و بنابرین فرم کلی جواب به شکل:

$$f(n) = a1^n+b2^n+c3^n$$

خواهد بود.
حال باید دستگاهی با سه معادله و سه مجهول تشکیل دهیم تا ثابت ها را به دست بیاوریم.
با توجه به اینکه فقط دو مقدار اولیه به ما داده شده است، میتوانیم ابتدا با استفاده از خود رابطه بازگشتی، مقدار سومی هم به دست بیاوریم تا دستگاه را تشکیل دهیم:

$$f(3) = 4f(2) - 3f(1) + 8 = 55$$

حال با حل دستگاه زیر، مقادیر ثابت ها را به دست میاوریم:

$$a+2b+3c = -1$$
$$a + 4b + 9c = 11$$
$$a + 8b + 27c = 55$$

و خواهیم داشت :

$$a = -3$$
$$b = -4$$
$$c = 10/3$$

در انتها با جایگذاری ثابت ها در فرم بسته رابطه بازگشتی، جواب را خواهیم داشت:

$$f(n) = -3 -(2^(n+2)) + 103^(n-1)$$

ب)

چنانچه در جایگاه n ام دنباله، رقم ۱ داشته باشیم، بقیه دنباله به $b_{(n-1)}$ حالت ساخته می‌شود و چنانچه رقم n ام ۰ باشد، رقم بعدی به ناچار باید ۱ باشد و در نتیجه بقیه دنباله به $b_{(n-2)}$ حالت ساخته میشوند. بنابرین طبق اصل جمع برای b_n رابطه بازگشتی زیر را خواهیم داشت:

$b_n = b_{(n-1)} + b_{(n-2)}$

که $b_1 = 2, b_2 = 3$ خواهد بود.

رابطه ذکرشده را میتوان با معادله مشخصه حل کرد :

$r^2 - r - 1 = 0$
$r_1 = (1+\sqrt{5})/2$
$r_2 = (1-\sqrt{5})/2$

بنابرین فرم بسته b_n به شکل زیر خواهد بود:

$b_n = a(r_1)^n + c(r_2^n)$

با استفاده از مقادیر b_1 , b_2 میتوانیم مقدار a , c را به دست آوریم که برابر با :

$a = (3+\sqrt{5})/(2\sqrt{5})$
$c = (5 - 3\sqrt{5})/(10)$

خواهند بود.

تمرین دوم: انادر پیچ سالو

الف)

```
void solve(int[] A, int l, int r) {
    if (l >= r)
        return;

    int sum = 0;
    for(int i = l; i <= r; i = i + 2) {
        sum += A[i];
    }

    int mid = (l + r) / 2;

    solve(A, l, mid);
    solve(A, mid + 1, r);
}
```

$$T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{2}$$
$$n = 2^k : T(2^k) = 2T(2^{k-1}) + 2^{k-1}$$
$$T(2^k) = F(k) \rightarrow F(k) = 2F(k-1) + 2^{k-1}$$

$$r^k = 2r^{k-1} \rightarrow r^{k-1}(r-2) = 0 \rightarrow r = 2$$
$$Fh(k) = C_1(2)^k$$
$$Fp(k) = C_2 \times k \times 2^k$$
$$F(k) = F_h(k) + F_p(K) = C_1 \times 2^k + C_2 \times k \times 2^k$$
$$T(n) = C_1 n + C_2 n \times log(n)$$
$$\text{time compelxity} = \mathrm{O}(\mathrm{n} \times \log(\mathrm{n}))$$

ب)

$$T(n) = 2T(\sqrt{n}) + \log n$$

$$n = 2^m \rightarrow T(2^m) = 2T(2^{m/2}) + m$$

$$w(m) = 2w(m/2) + m$$

$$\xrightarrow{\text{master}} \quad w(m) = \theta(m \times \log m)$$

$$n = 2^m \rightarrow m = \log n$$

$$\rightarrow \boxed{T(n) = \theta(\log n \times \log \log n)}$$

تمرین سوم: ۲۰۶ آلبالویی

با توجه به اینکه رقم ۹ نمی‌تواند در عدد استفاده شود، برای جایگاه اول در اعداد ۸ حالت (ارقام به جز ۰ و ۹) و برای جایگاه های بعدی در صورت وجود، ۹ حالت (ارقام ۰ تا ۸) داریم و طبق اصل ضرب جواب برای عددی x رقمی، جواب برابر با ۸*(۹^(x-1)) خواهد بود.
اگر بخواهیم با استفاده از توان عادی و یک حلقه جواب را محاسبه کنیم، کد ما پیچیدگی زمانی از مرتبه O(x) خواهد داشت که با توجه به محدوده x پیچیدگی مناسبی نیست. (چرا؟)
با به کارگیری روش تقسیم و غلبه در این سوال می‌توانیم با تقسیم مسئله محاسبه توان به زیر مسئله و سپس ترکیب راه حل‌ها به راهی از مرتبه O(logx) برسیم.
نحوه تقسیم به زیر مسئله و ترکیب با استفاده از این نکته خواهد بود که:

power(a,x) = if(x%2==1) : power(a,x/2)*power(a,x/2)*a
if(x%2==0) : power(a,x/2)*power(a,x/2)

البته باید توجه شود که در صورت ذخیره نکردن جواب از پیش محاسبه شده و دو بار محاسبه عبارت power(a,x/2)، راه حل همچنان از O(x) خواهد بود.در کد راه حل توابعی برای جمع و ضرب نوشته شده اند، هنگامی که در پیاده سازی به عمل باقی مانده نیاز داریم، خوب است که این توابع را برای کار راحت‌تر پیاده‌سازی کنیم.

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll mo = 1e9+7;

ll add(ll a,ll b){
  return (a+b)%mo;
}

ll mul(ll a,ll b){
  return (a*b)%mo;
}

ll po(ll a,ll b){
  if(b==0)return 1;
  ll ans = po(a,b/2);
  return b%2 ? mul(a,mul(ans,ans)):mul(ans,ans);
}
int main(){
  ll t;
  cin >> t;
  while(t--){
    ll d;
    cin >> d;
    cout<<mul(8,po(9,d-1))<<endl;
  }
  return 0;
}
```

تمرین چهارم: Halo

```cpp
#include<bits/stdc++.h>
using namespace std;
int n;
int partion(int l,int h,int arr[]){
   int pivot=arr[l];
   int i=l,j=h;
   while(i<j){
      do{
         i++;
      }while(arr[i]<=pivot && i<n);
      do{
         j--;
      }while(arr[j]>pivot && j>0);
      if(i<j) swap(arr[i],arr[j]);
   }
   swap(arr[j],arr[l]);
   return j;
}
void quicksort(int l,int h,int arr[]){
   if(l<h){
      int j=partion(l,h,arr);
      quicksort(l,j,arr);
      quicksort(j+1,h,arr);
   }
}
int main(){
   cin>>n;
   int *arr=new int[n];
   for(int i=0;i<n;i++) cin>>arr[i];
   quicksort(0,n,arr);
   if(n%2==0) cout<<(arr[n/2]+arr[n/2-1])/2<<endl;
   else cout<<arr[n/2]<<endl;
   return 0;
}
```