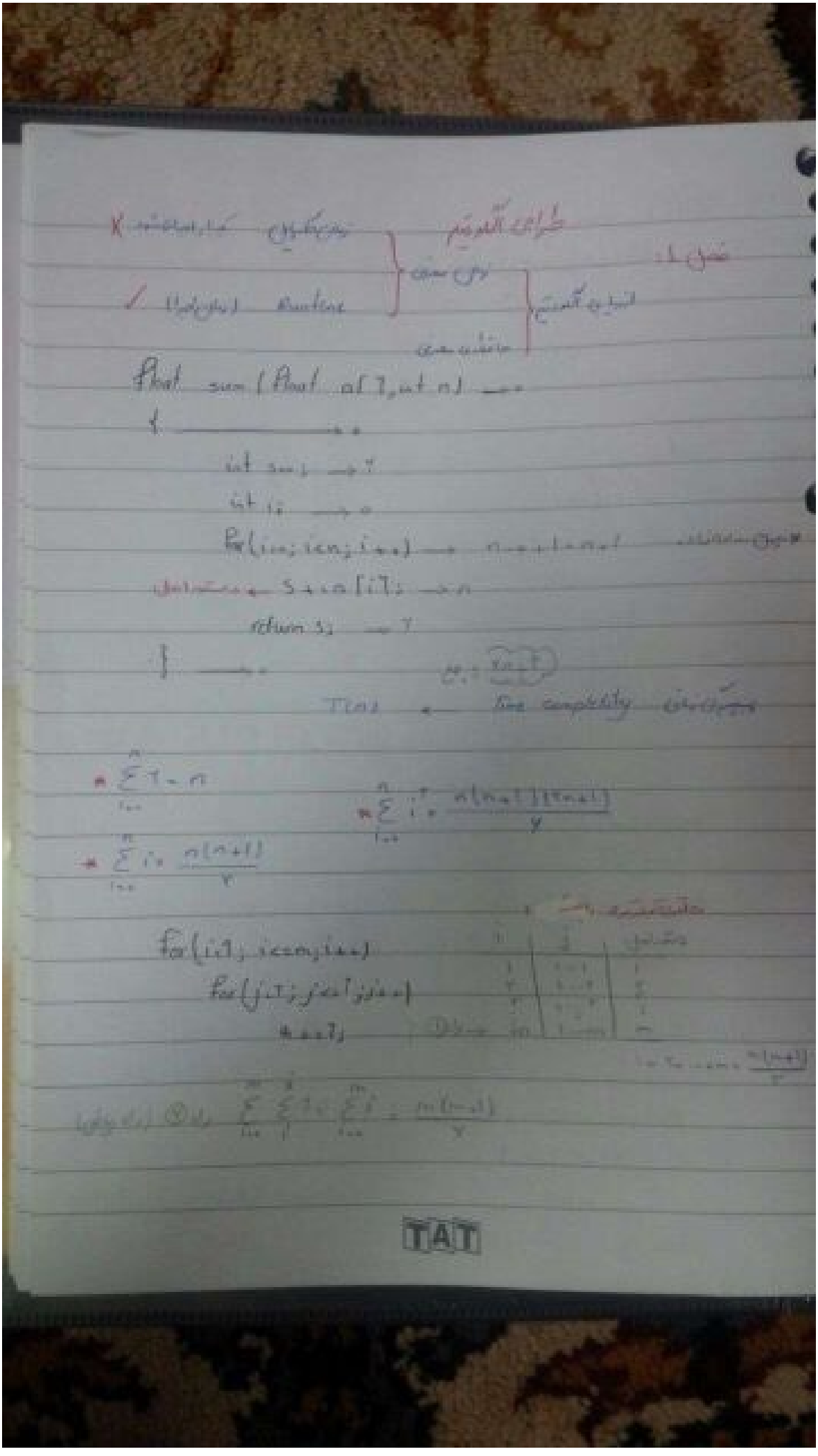
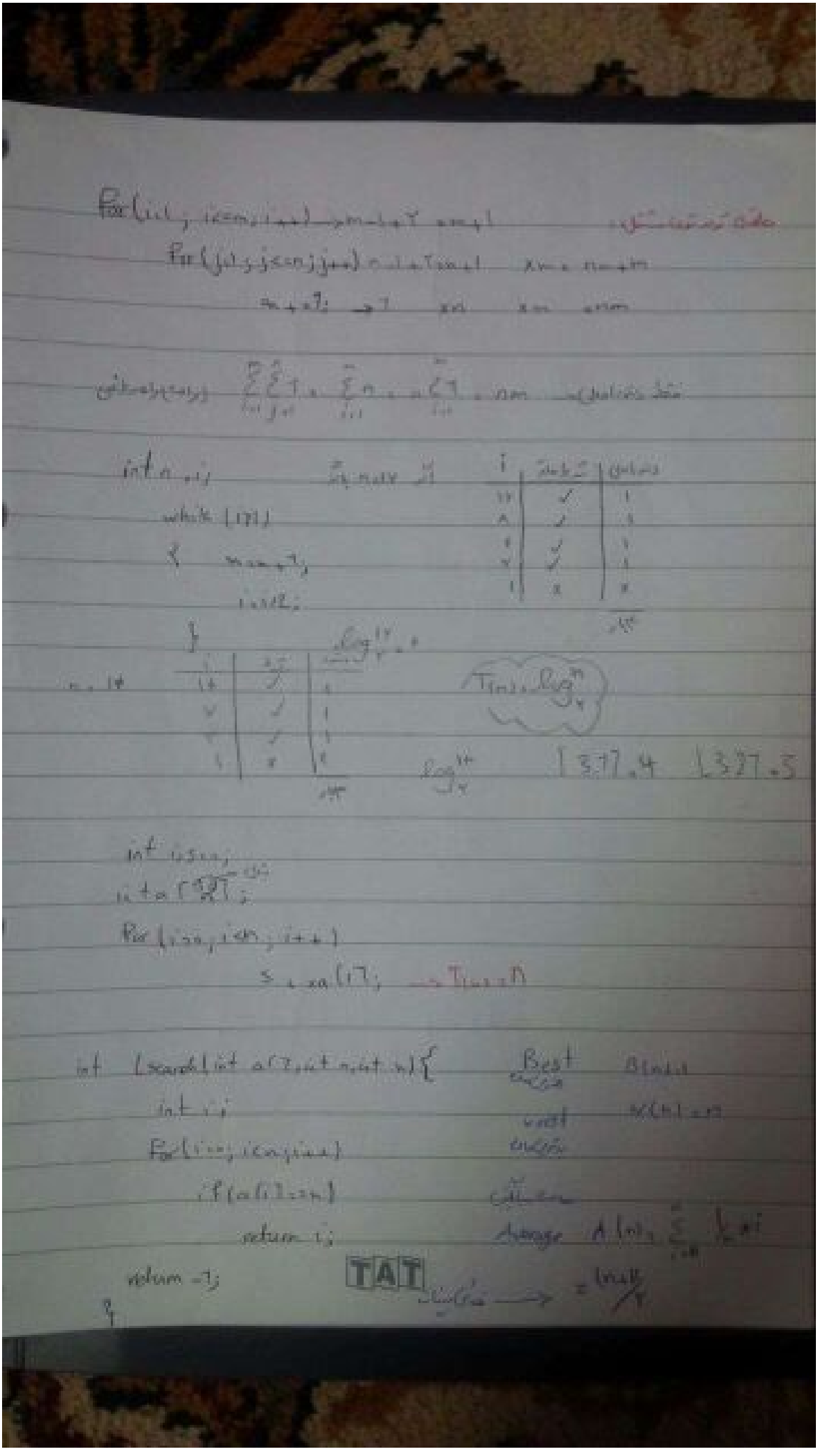
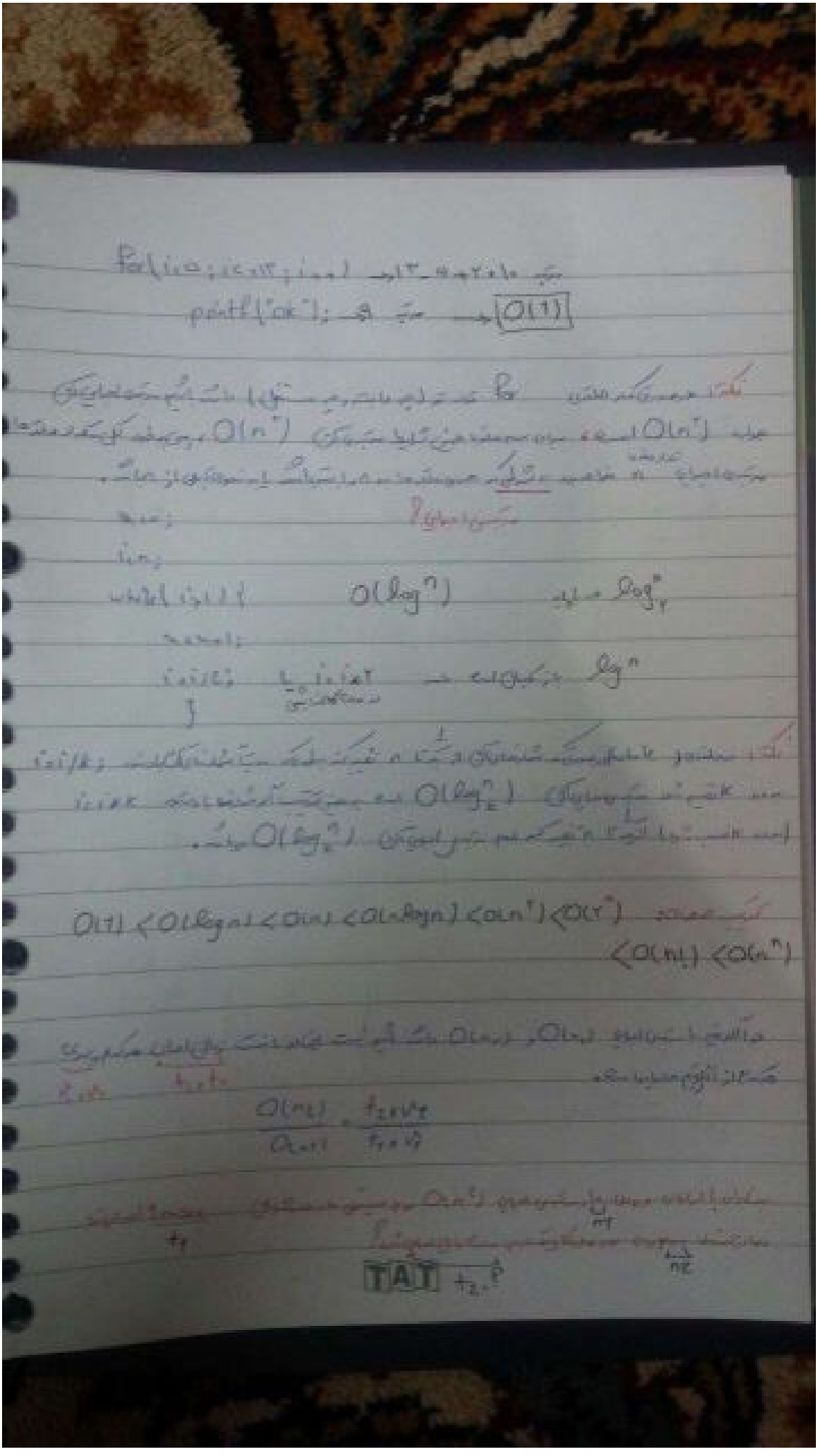
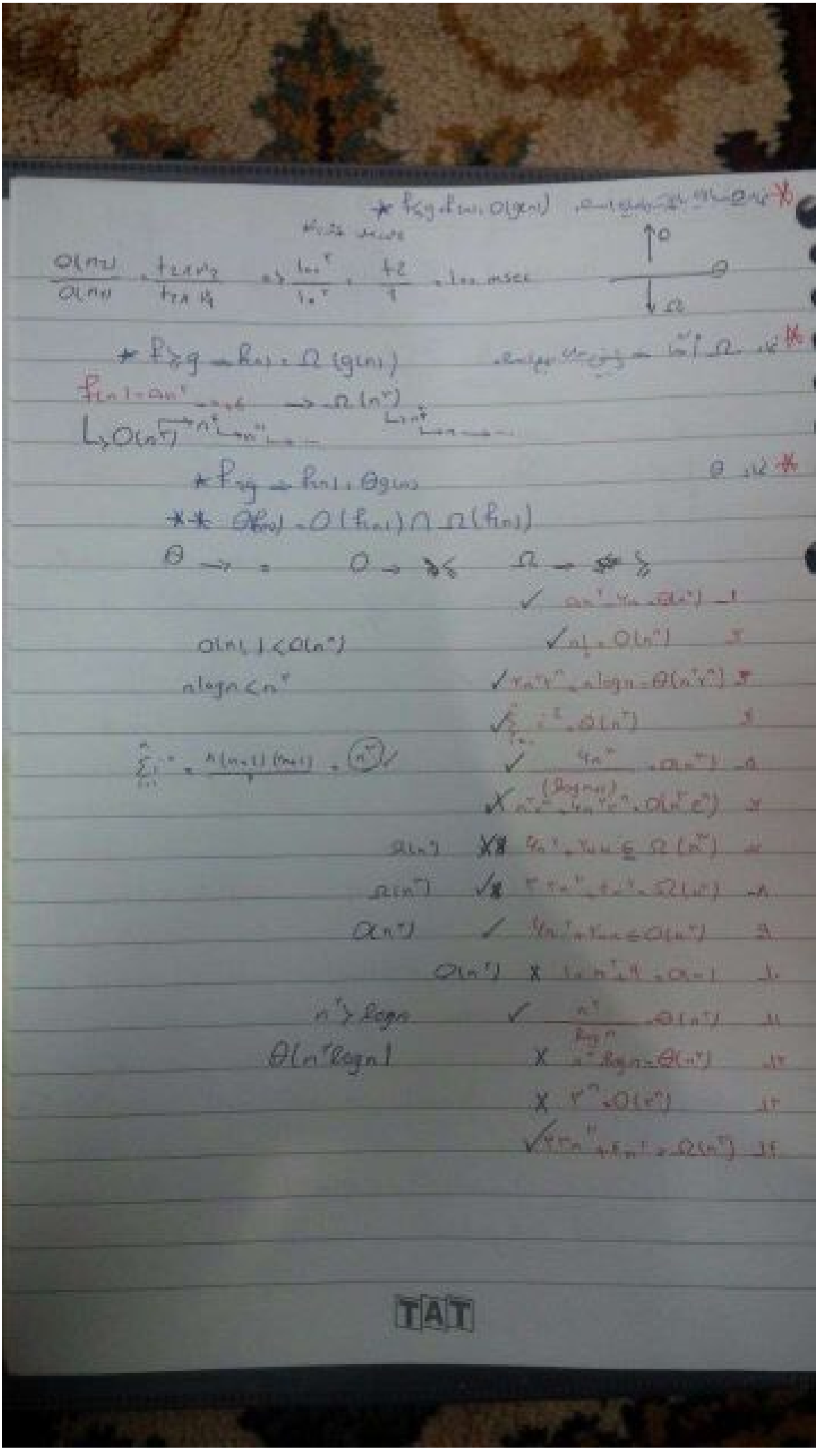
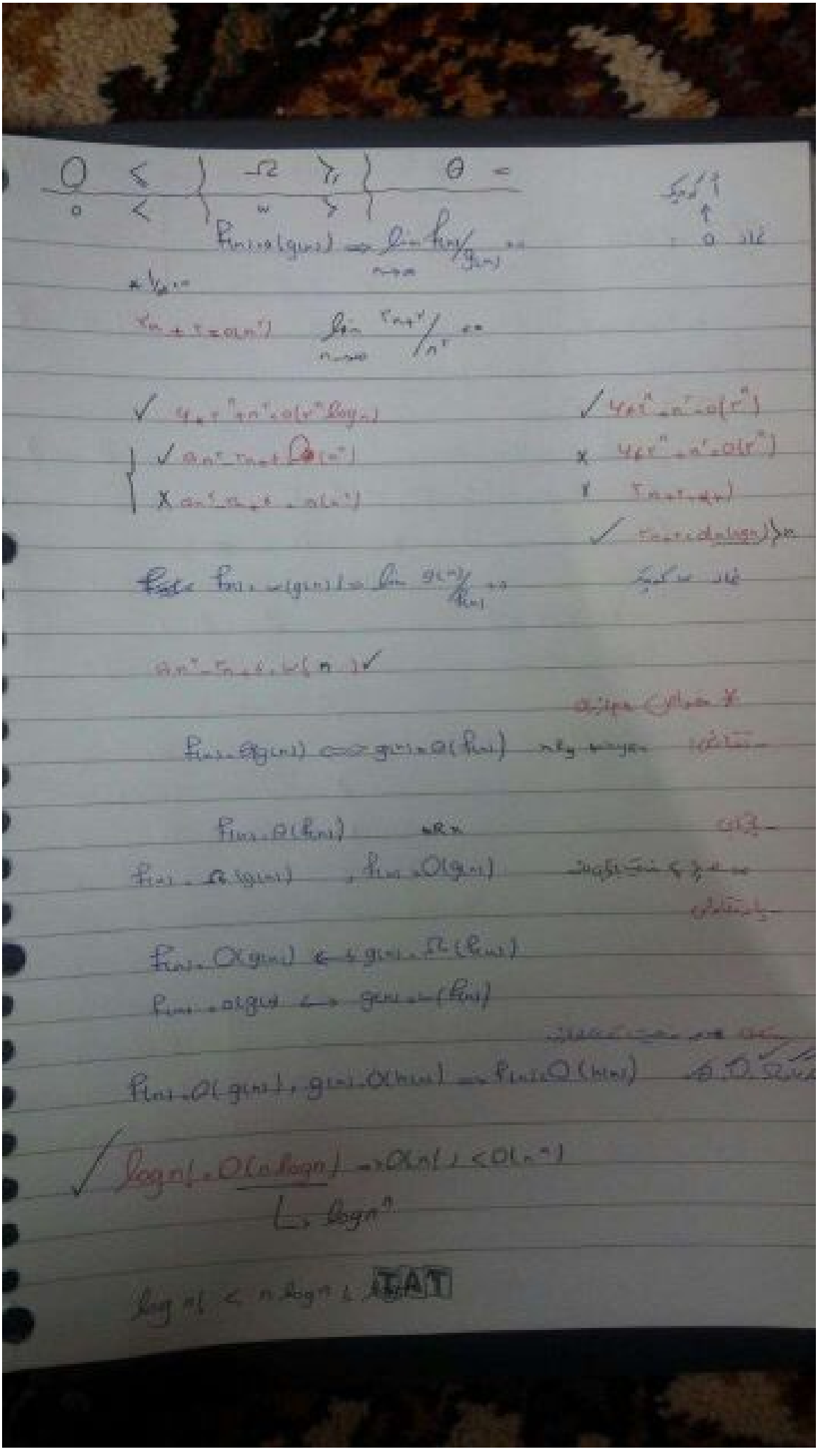
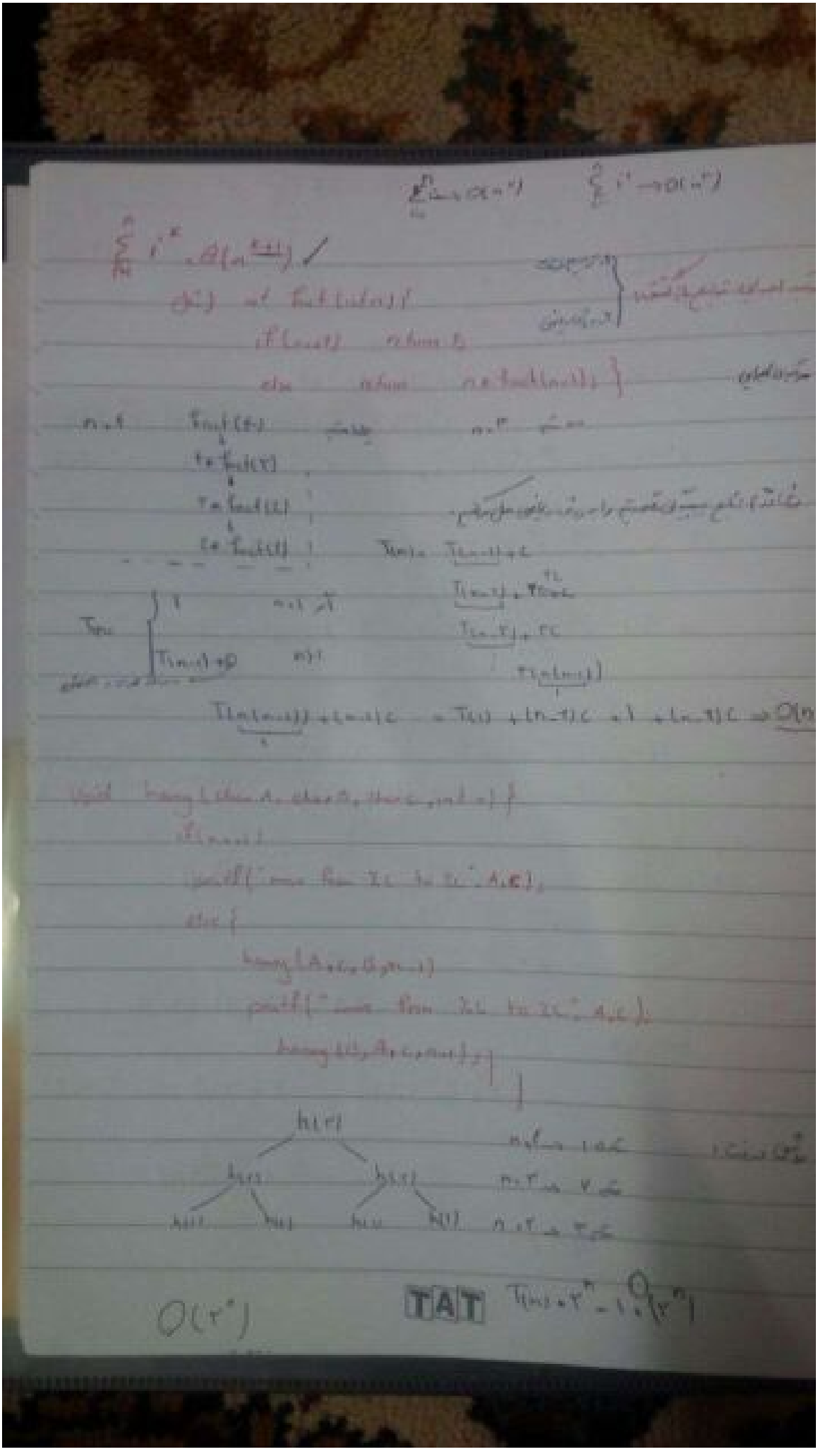


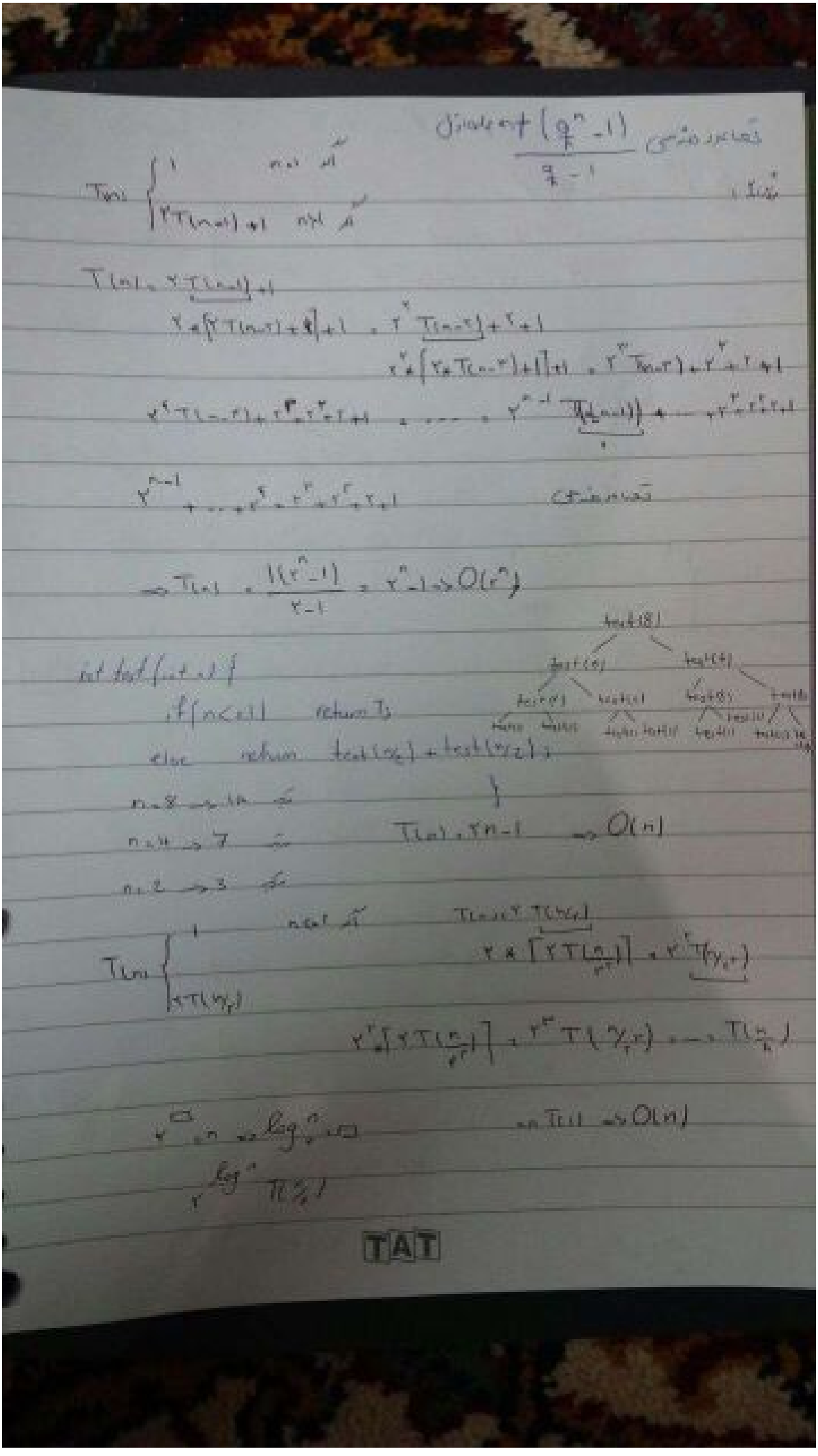






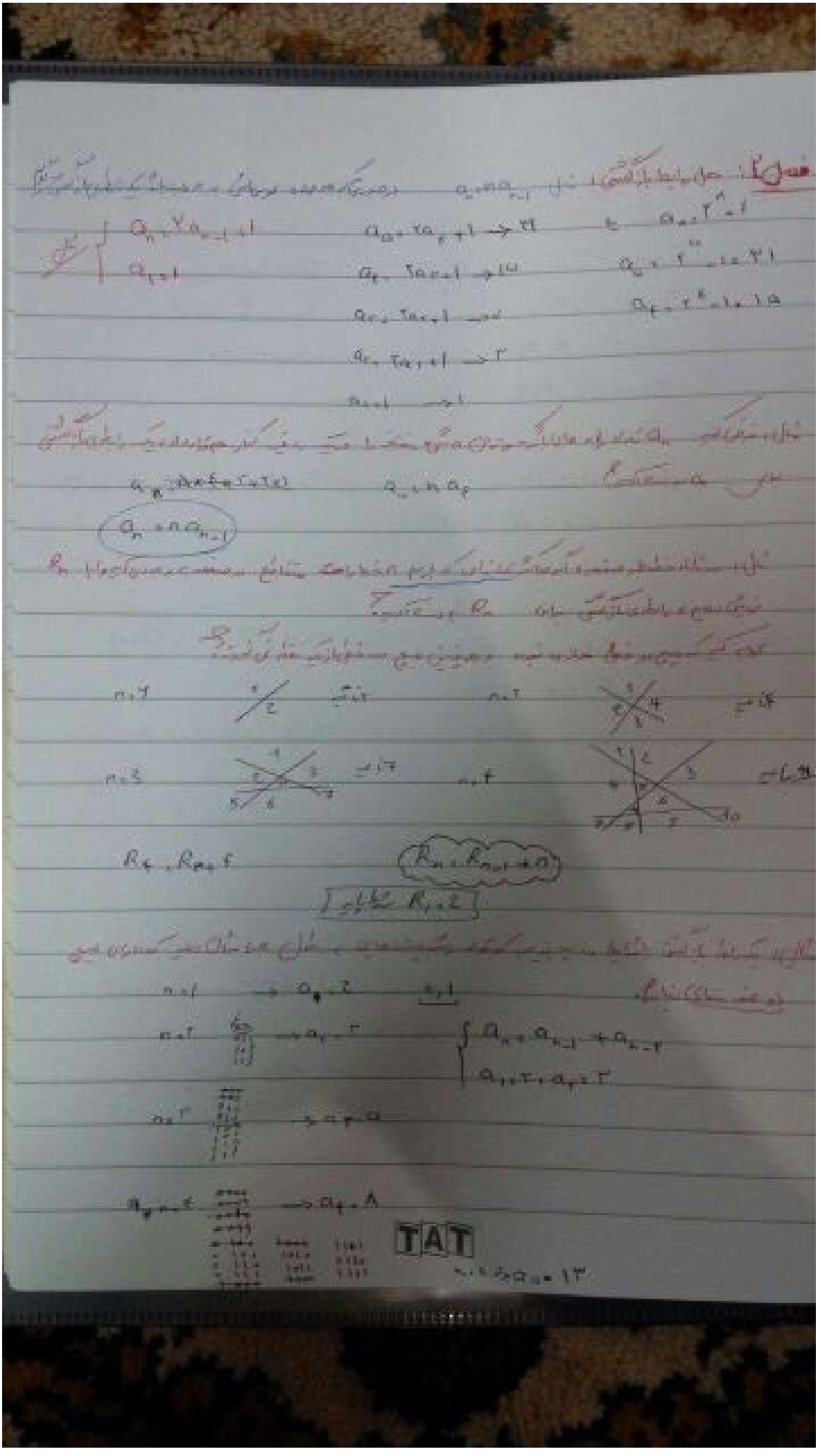






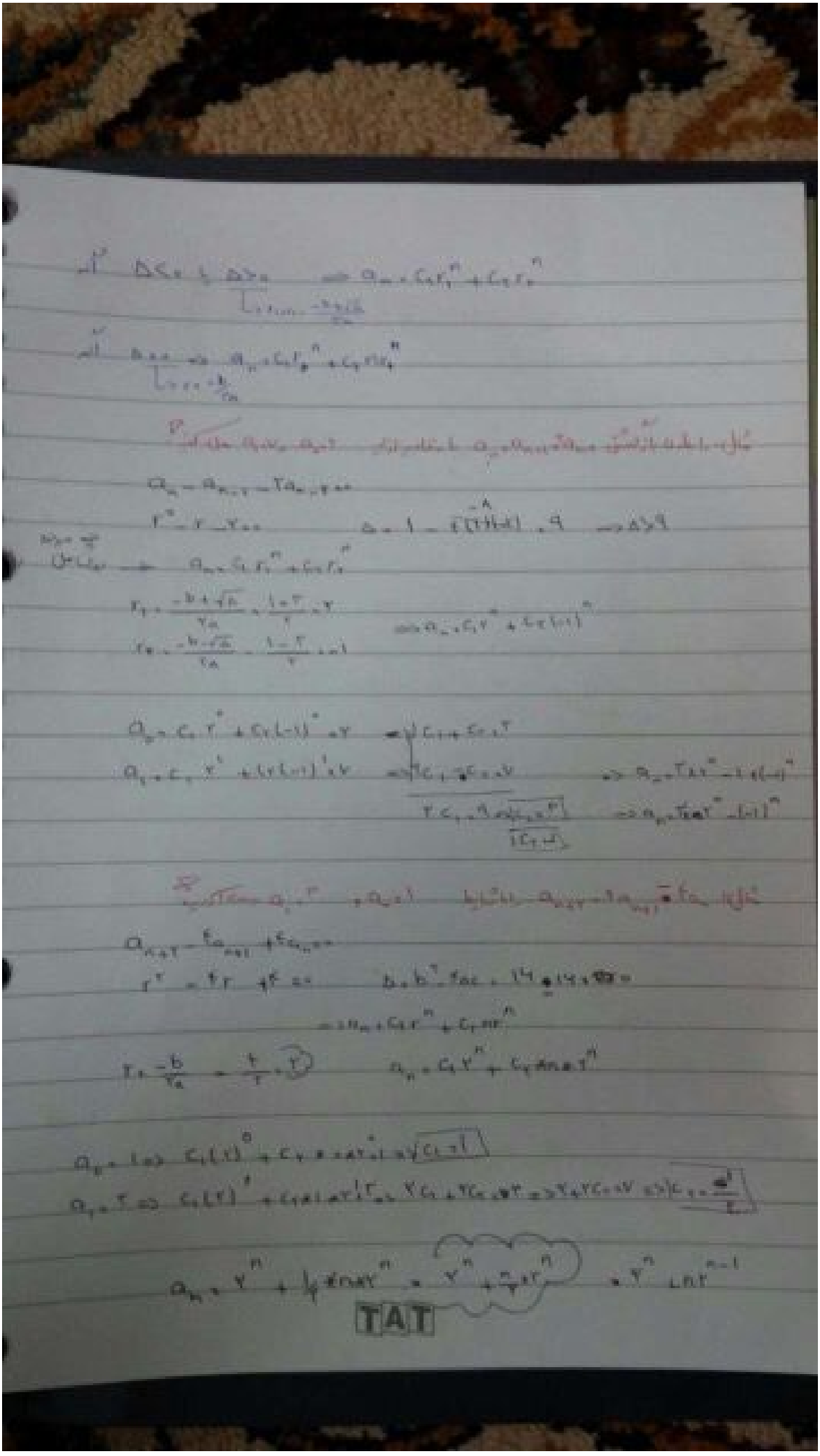
TAT

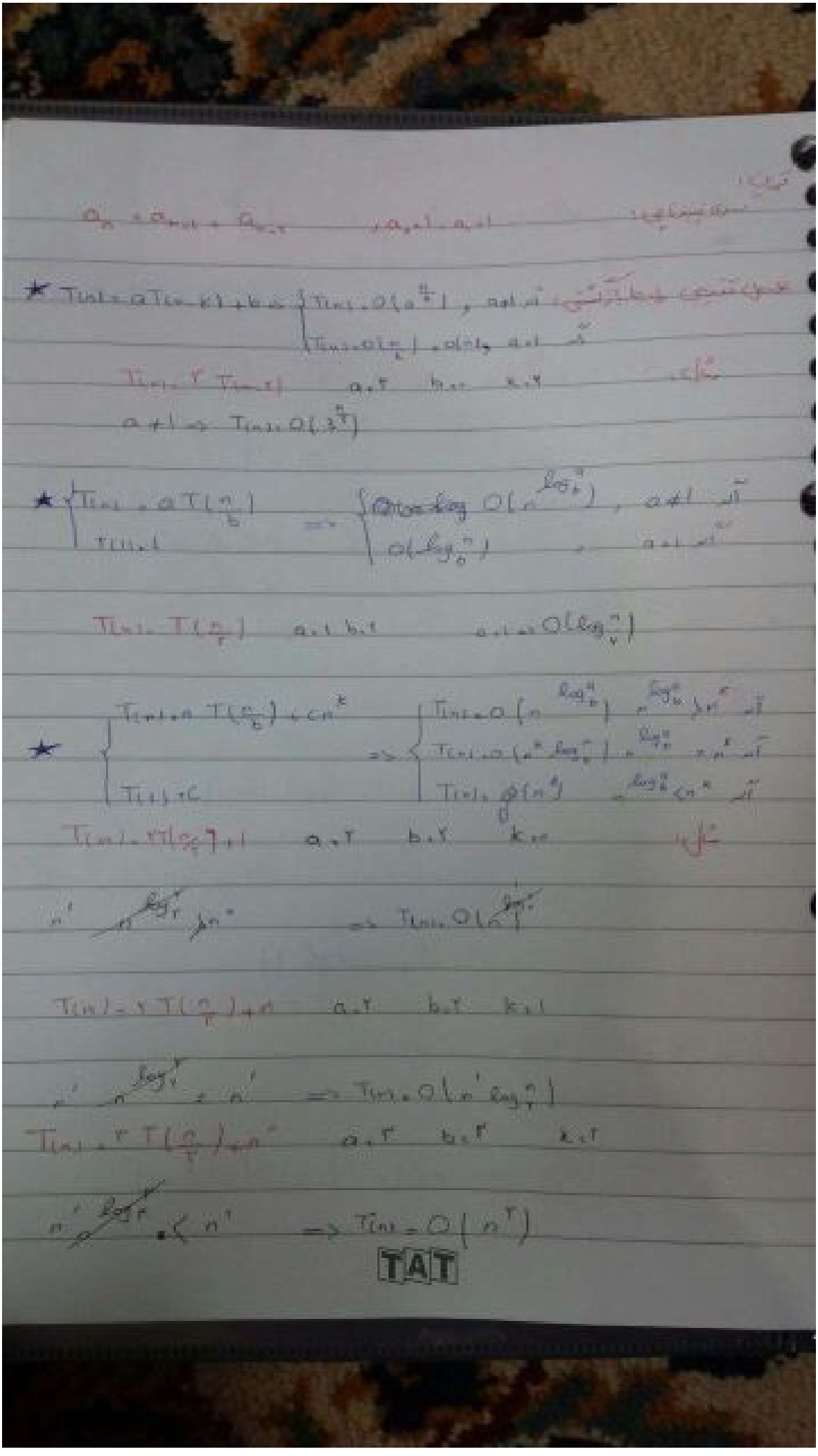
Scanned by CamScanner

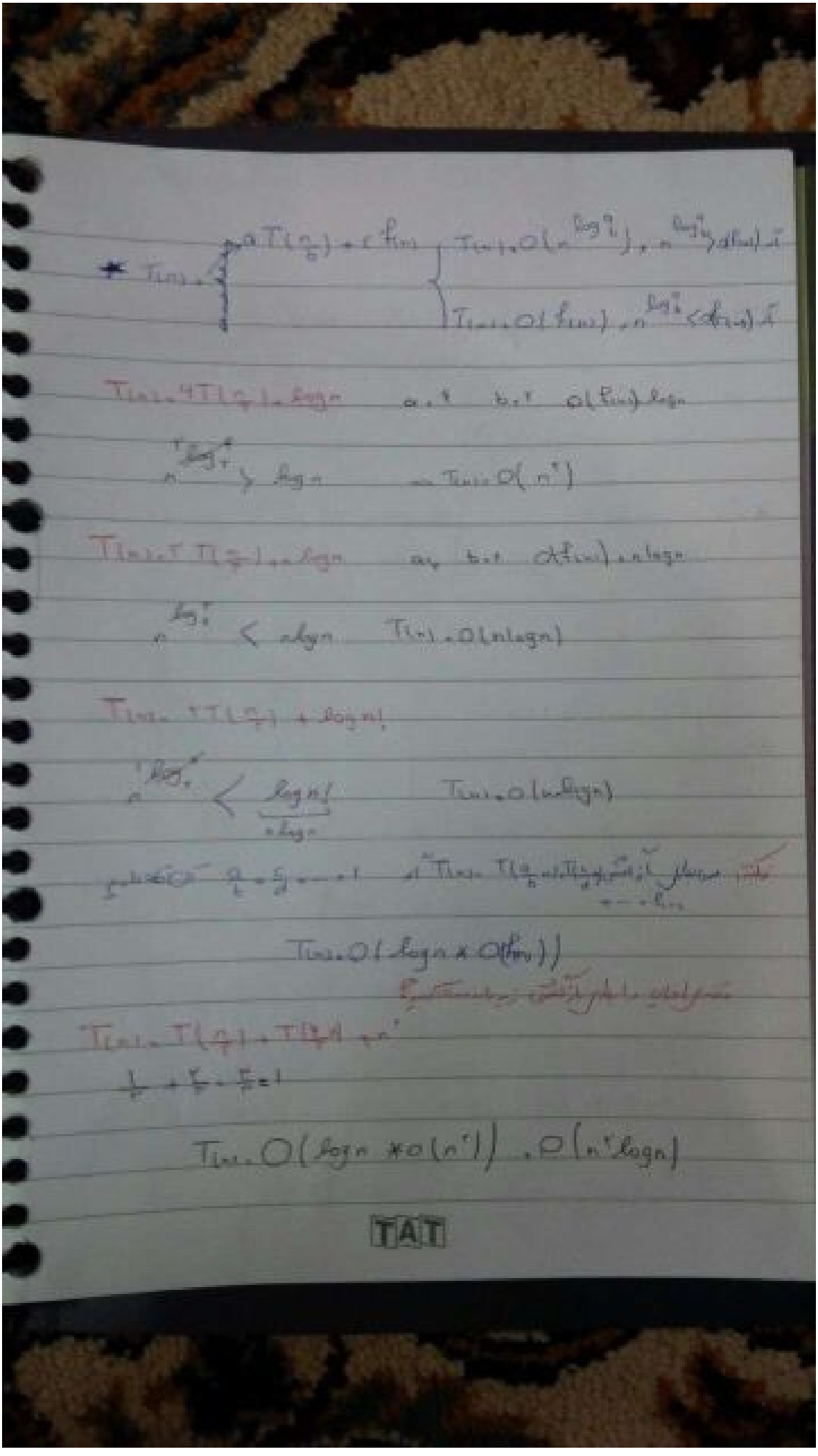


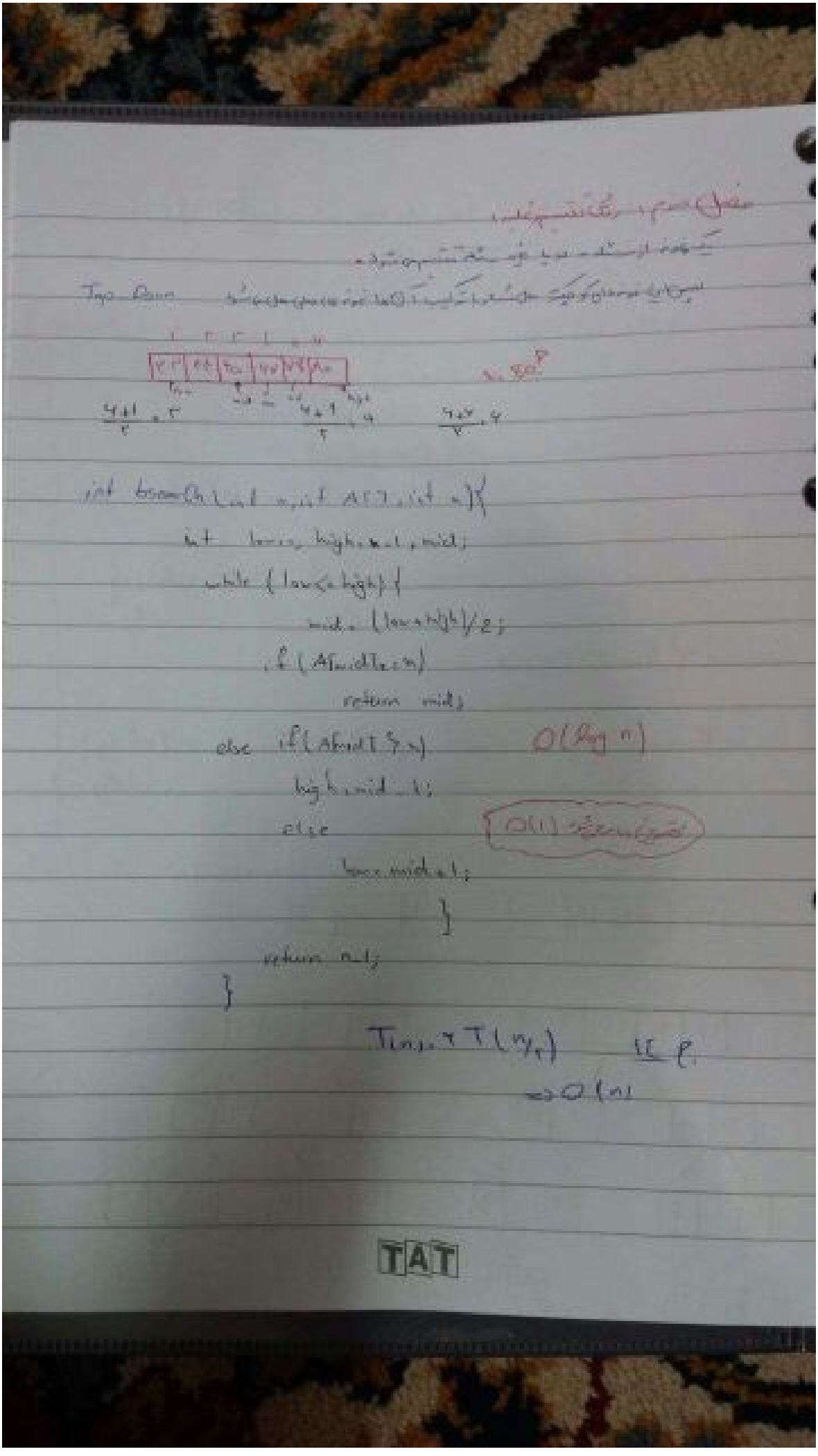
Scanned by CamScanner

Scanned by CamScanner









```
int binarySearch(int arr[], int n){
    int low, high, mid;
    while(low <= high){
        mid = (low + high)/2;
        if(arr[mid] == n)
            return mid;
        else if(arr[mid] > n)
            high = mid - 1;
        else
            low = mid + 1;
    }
    return -1;
}
```

O(log n)

O(1)

Time = T(n/2) + C

=> O(log n)

TAT

Scanned by CamScanner

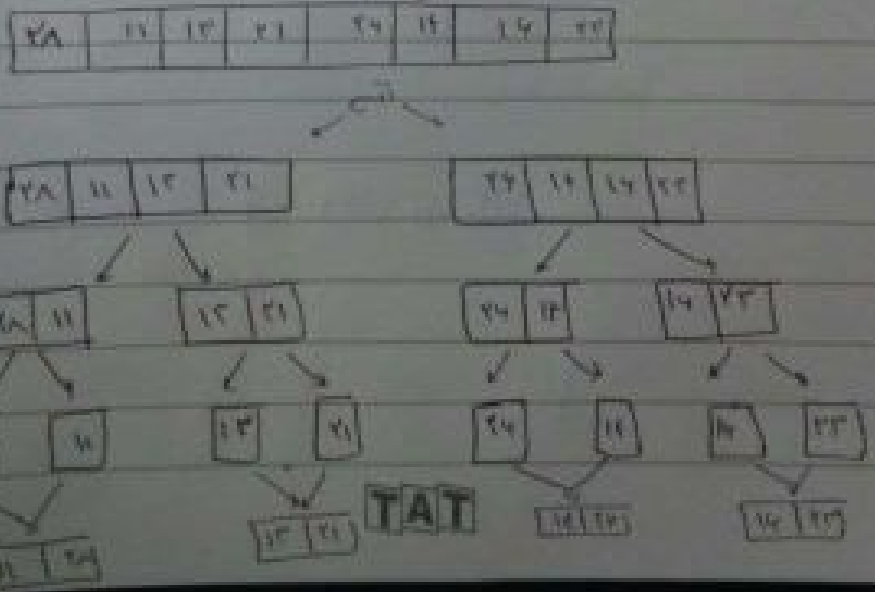
```
int bsearch(int low, int high, int x, int A[]) {
  int mid;
  if (low > high) return -1;
  else {
    mid = (low+high)/2;
    if (A[mid] == x)
      return mid;
    else if (A[mid] > x)
      return bsearch(A, mid-1, low, x);
    else
      return bsearch(A, high, mid+1, x); }
  }
```

$$T(n) = \begin{cases} b & n=1 \\ T(\frac{n}{b}) + 1 & n>1 \end{cases}$$

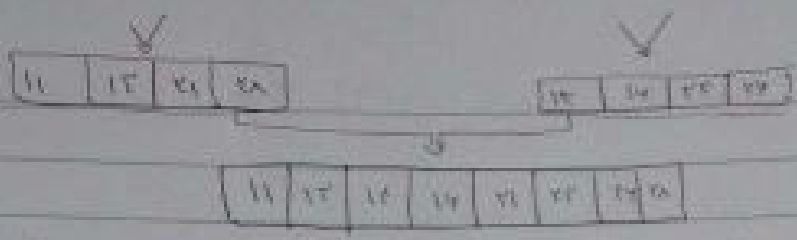
$T(n) = aT(\frac{n}{b}) + cn^d \rightarrow$ Big O $n^d \Rightarrow O(n^d \log_b^n)$

a=1 b=2 ... $\log_2^1 = 0$ $\Rightarrow O(\log n)$

(merge sort)

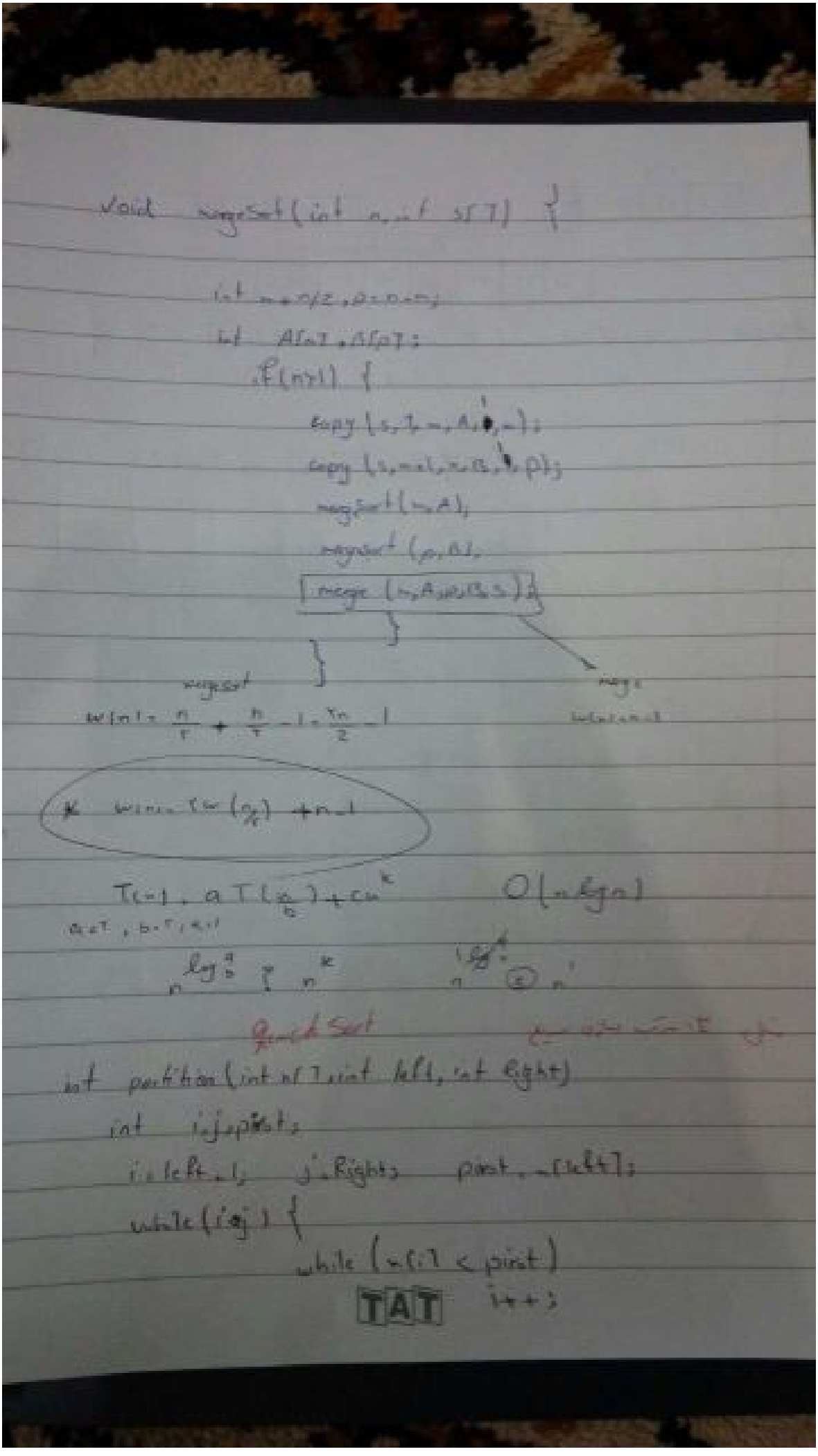


Scanned by CamScanner



```
Avoid merge(int arr1[], int n1, int arr2[], int n2, int arr3[]) {
    int i, j, k;
    i = 0; j = 0; k = 0;
    while(i < n1 && j < n2) {
        if(arr1[i] < arr2[j]) { arr3[k] = arr1[i]; i++; }
        else { arr3[k] = arr2[j]; j++; }
        k++;
    }
    if(i < n1)
        while(i < n1) { arr3[k] = arr1[i]; i++; k++; }
    else if(j < n2)
        while(j < n2) { arr3[k] = arr2[j]; j++; k++; }
}
```

TAT



Scanned by CamScanner

while(a[j] > pivot) j--;

if (i<j) swap(a[i],a[j]); }

swap(a[left],a[j]);

return j; }

Time and O(n)

void QuickSort(int a[], int left, int right) {

int pivotpoint;

if (left < right){

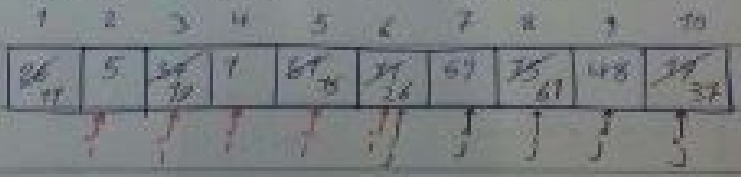
pivotpoint = partition(a, left, Right);

QuickSort(a, left, pivotpoint-1);

QuickSort(a, pivotpoint+1, right); }

}



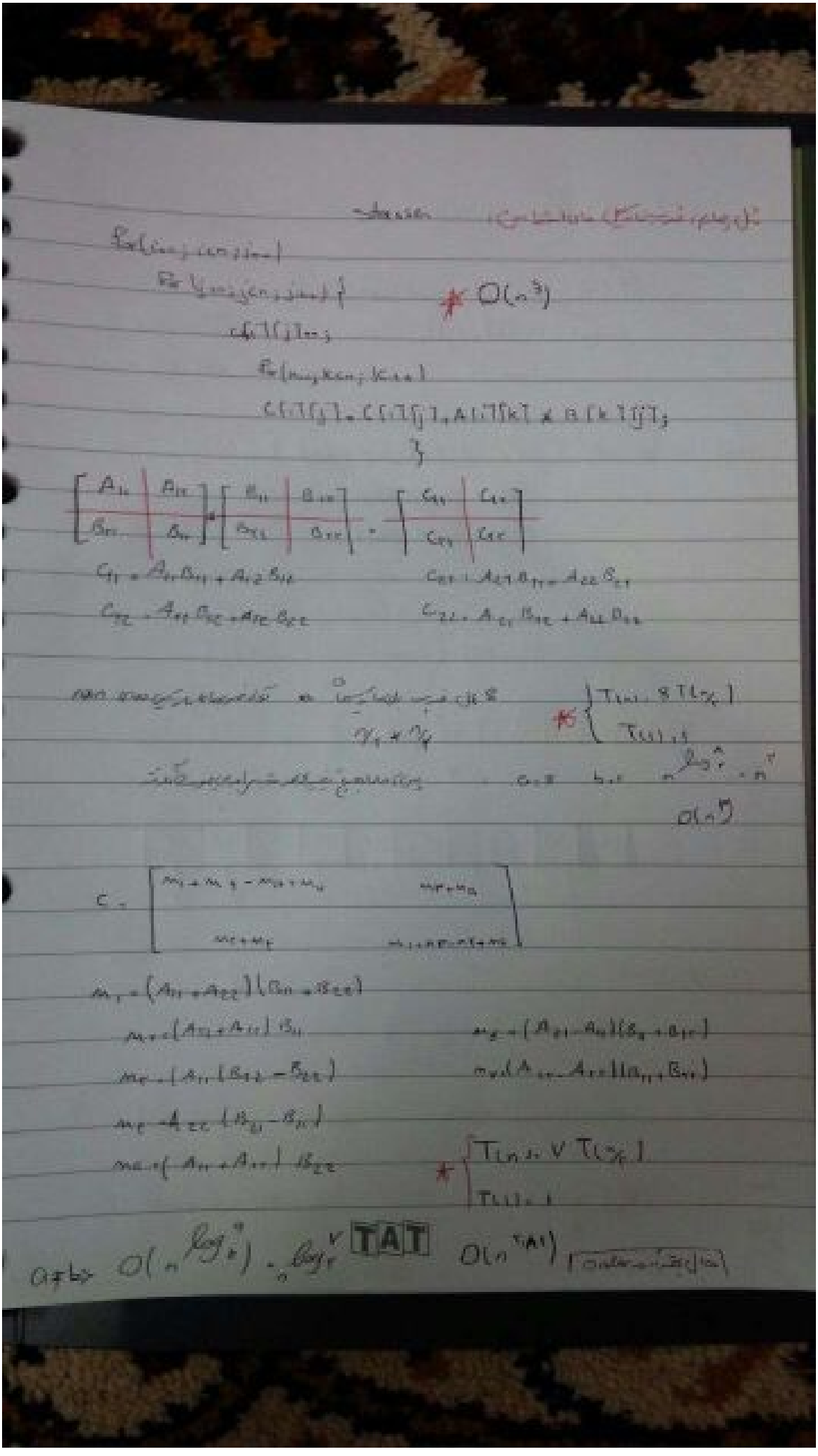


QuickSort

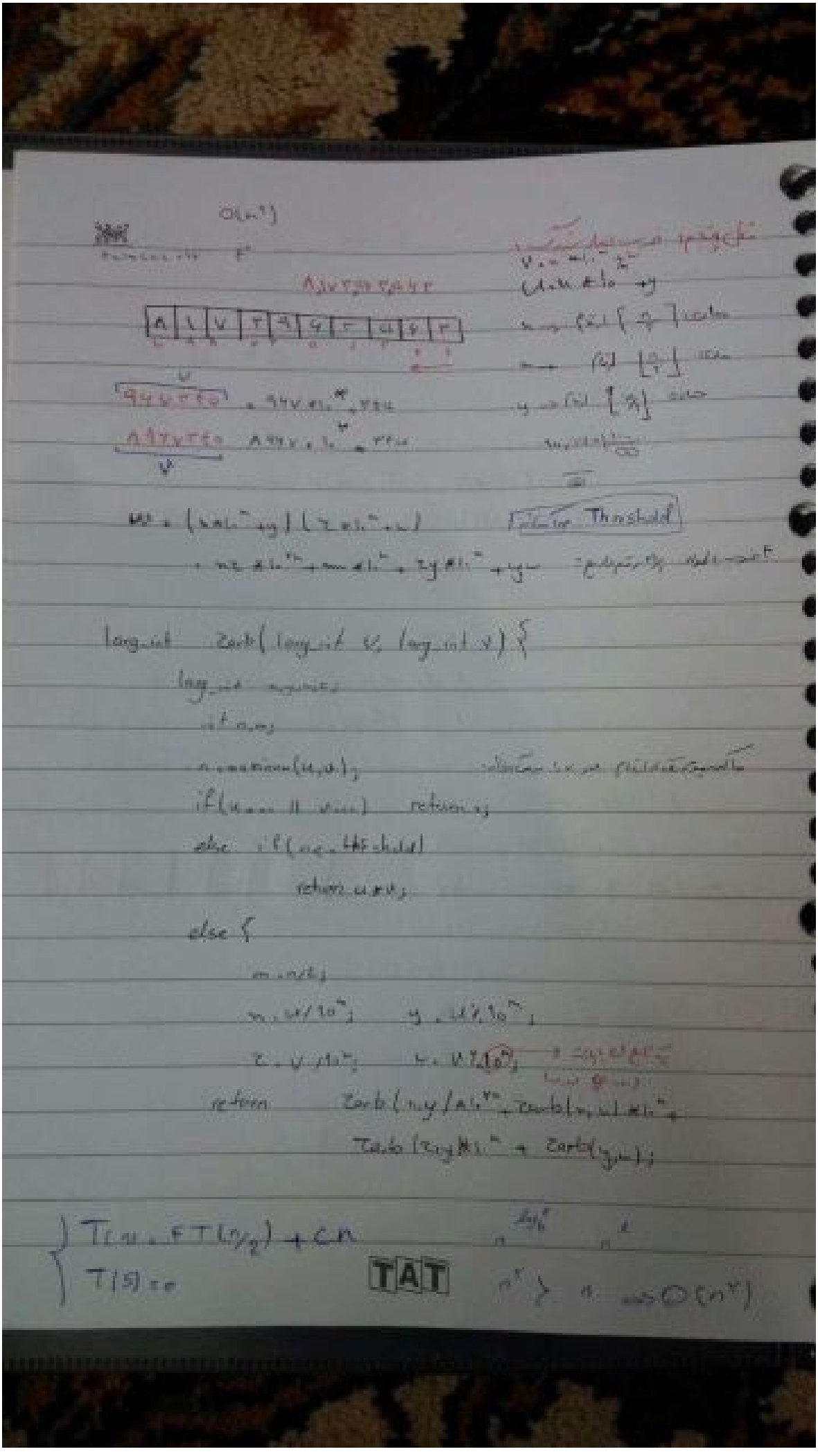
→ Θ(n log n)

TAT

O(n)

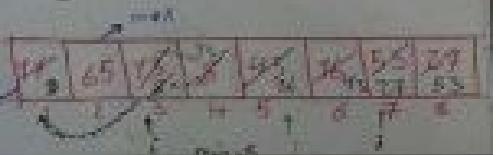


Scanned by CamScanner



Scanned by CamScanner

```
MAX = s[0];
for(i=1; i<n; i++)
    if(s[i] > MAX)
        MAX = s[i];
max = s[0];
min = s[0];
for(i=1; i<n; i++) { if(s[i] < min) min = s[i];
                     else if(s[i] > max) max = s[i]; }
if (s[1] > s[2]) {max = s[1]; min = s[2];}
else {min = s[1]; max = s[2];}
for(i=3; i<n-1; i+=2){
    if(s[i] > s[i+1])
        swap(s[i], s[i+1]);
    if (s[i] < min)
        min = s[i];
    if (s[i+1] > max)
        max = s[i+1]; }
```



TAT

Scanned by CamScanner

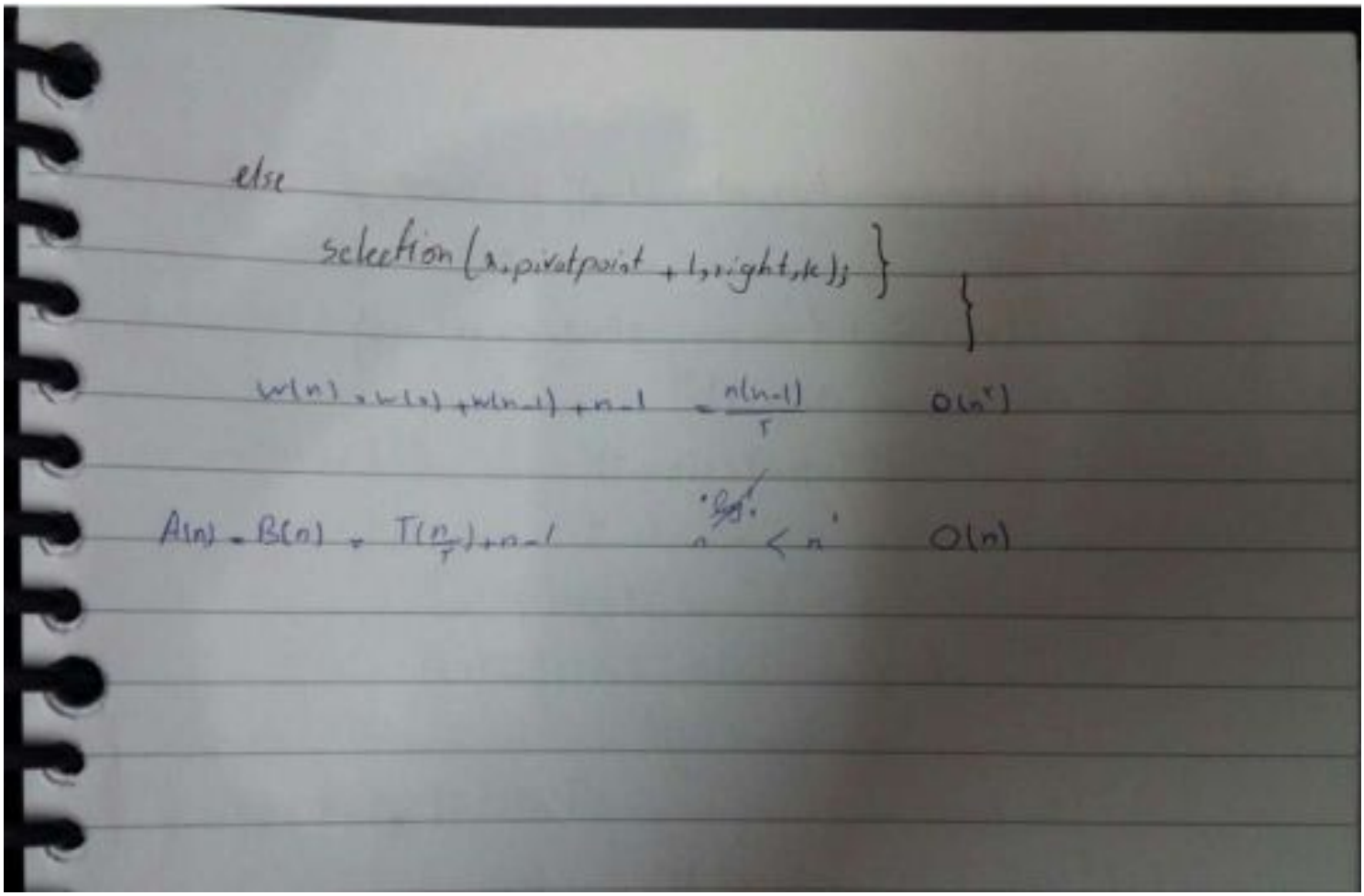
```
void find(int low,int high,int s[],int min,int max){
int mid;
if(low==high) { max=min=s[low]; return;}
else if(low==high-1){
    if(s[low]<s[high]){
        min=s[low];
        max=s[high]; }
    else { min=s[high]; max=s[low];}
    return; }
mid=(low+high)/2;
find(low,mid,s[],min1,max1);
find(mid+1,high,s[],min2,max2);
max=maximum(max1,max2);
min=minimum(min1,min2); }
```

Selection

```
int selection(int a[],int left,int right,int k){
int pivotpoint;
if(left==right) return a[left];
else{
    pivotpoint=partition(a,left,right);
    if(k==pivotpoint)
        return a[pivotpoint];
    else if(k<pivotpoint)
        selection(a,left,pivotpoint-1,k);
```

TAT

Scanned by CamScanner



Scanned by CamScanner

T(n) = T(n-1) + 3

[T(n-2) + 3] + 3 = T(n-2) + 3×2

T(n-(n-1)) = T(n-1) + 3 = 1 + 3(n-1) = 3n-2

```
int Fib(int n){
    if(n==1||n==2) return n;
    else
        return (Fib(n-1)+Fib(n-2));
}
```

O(2^n)

```
int fib(int n){
    int i, a[n];   a[1]=1; a[2]=1;

    for(i=3;i<=n;i++)
        a[i]=a[i-1]+a[i-2];
    return a[n];
}
```

O(n)

```
int fib(int n){
    int c,a=1,b=1,i;
    for(i=2;i<n;i++)
        c=a+b;
        a=b;
        b=c; }
    return c;
}
```

O(n)

TAT

Scanned by CamScanner

$(a+b)^n$ ...

$(a+b)^n = \binom{n}{0}a^n b^0 + \binom{n}{1}a^{n-1}b^1 + \binom{n}{2}a^{n-2}b^2 + \dots + \binom{n}{n}a^0 b^n$

$\binom{n}{k} = \binom{n}{n-k}$ ...

$\binom{n}{1} = n$

$\binom{n}{k} = C(n,k) = \frac{n!}{k!(n-k)!}$ ...

$\binom{n}{n} = \binom{n}{0} = 1$

$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$ ...

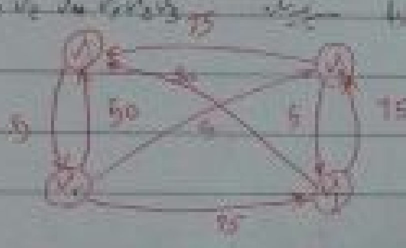
```
int tarkib(int n, int k){
    if(k==0 || k==n) return 1;
    else return tarkib(n-1,k) + tarkib(n-1,k-1);
}
```

B	0	1	2	3	4
0	1				
1	1	1			
2	1	2	1		
3	1	3	3	1	
4	1	4	6	4	1

TAT

```
int tarkib(int n, int k){
    int i,j, B[n][k];
    for(i=0; i<=n; i++){
        for(j=0; j<=min(i,k); j++){        O(nk)
            if(j==0 || i==j)
                B[i][j]=1;
            else
                B[i][j]= B[i-1][j-1]+ B[i-1][j]; } }
*   return B[n][k]; }
```

[illegible] Floyd → $O(n^3)$ [illegible]



$D_0 =$

	v_1	v_2	v_3	v_4
v_1	0	5	∞	∞
v_2	50	0	15	5
v_3	30	∞	0	15
v_4	15	∞	5	0

$D_1[i,j] = \min(D_0[i,j],\ D_0[i,1] + D_0[1,j])$

$D_k[i,j] = \min(D_{k-1}[i,j],\ D_{k-1}[i,k] + D_{k-1}[k,j])$

[illegible] → $D_0[i,j] = W[i,j]$ [illegible] c_{ij} / ∞ [illegible] / 0 [illegible]

$D_0[2,2] = \min(D_0[2,2], D_0[2,1] + D_0[1,2])$

$= \min(5, \infty + 5)$

$D_1[2,3] = \min(D_0[2,3], D_0[2,1] + D_0[1,3])$

$= \min(15, 5 + 10)$

بقیہ حل کریں

$\Rightarrow D_4 = \begin{bmatrix} 0 & 8 & 10 & 11 \\ 20 & 0 & 12 & 13 \\ 30 & 28 & 0 & 15 \\ 35 & 33 & 45 & 0 \end{bmatrix}$

~~$D_1[0,1] = \min(D_0[0,1], D_0[0,1] + D_0[1,1])$~~

~~$= \min(0, \infty + 0)$~~

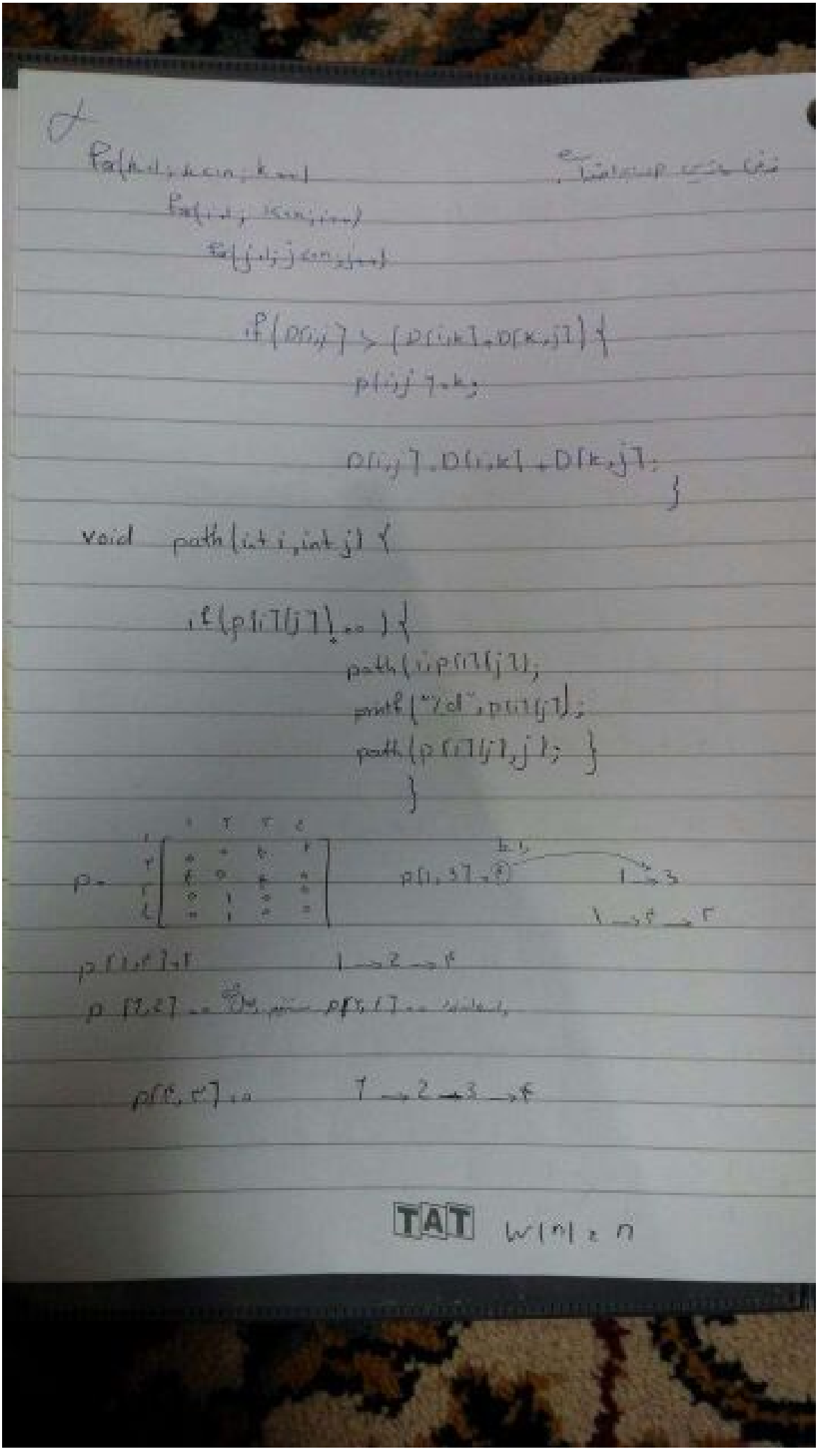
$D_1[1,3] = \min(\infty, 8 + 10)$

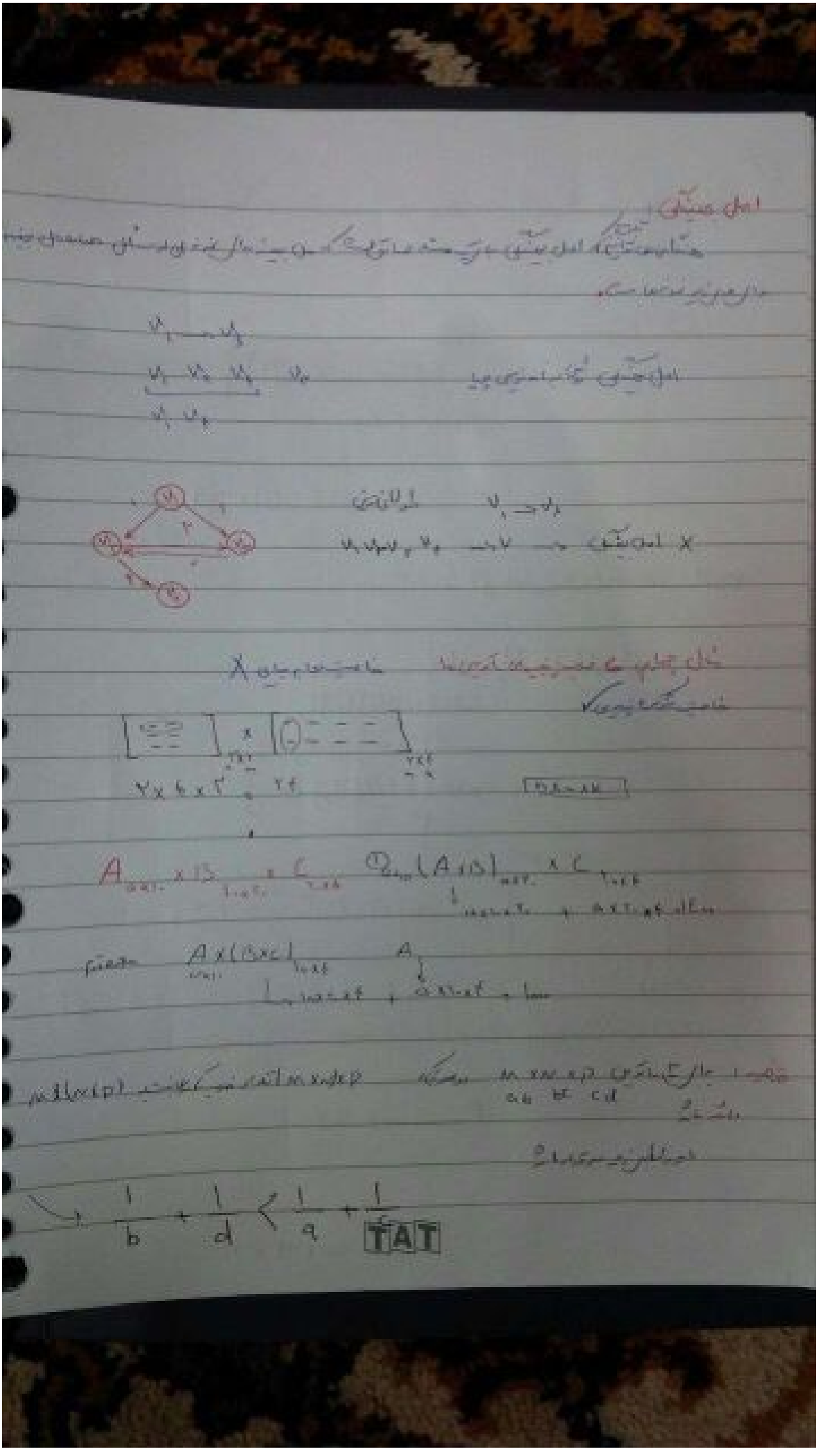
```
For(k=1; k<=n; k++)
    For(i=1; i<=n; i++)      } ⇒ O(n^3)
        For(j=1; j<=n; j++)
```

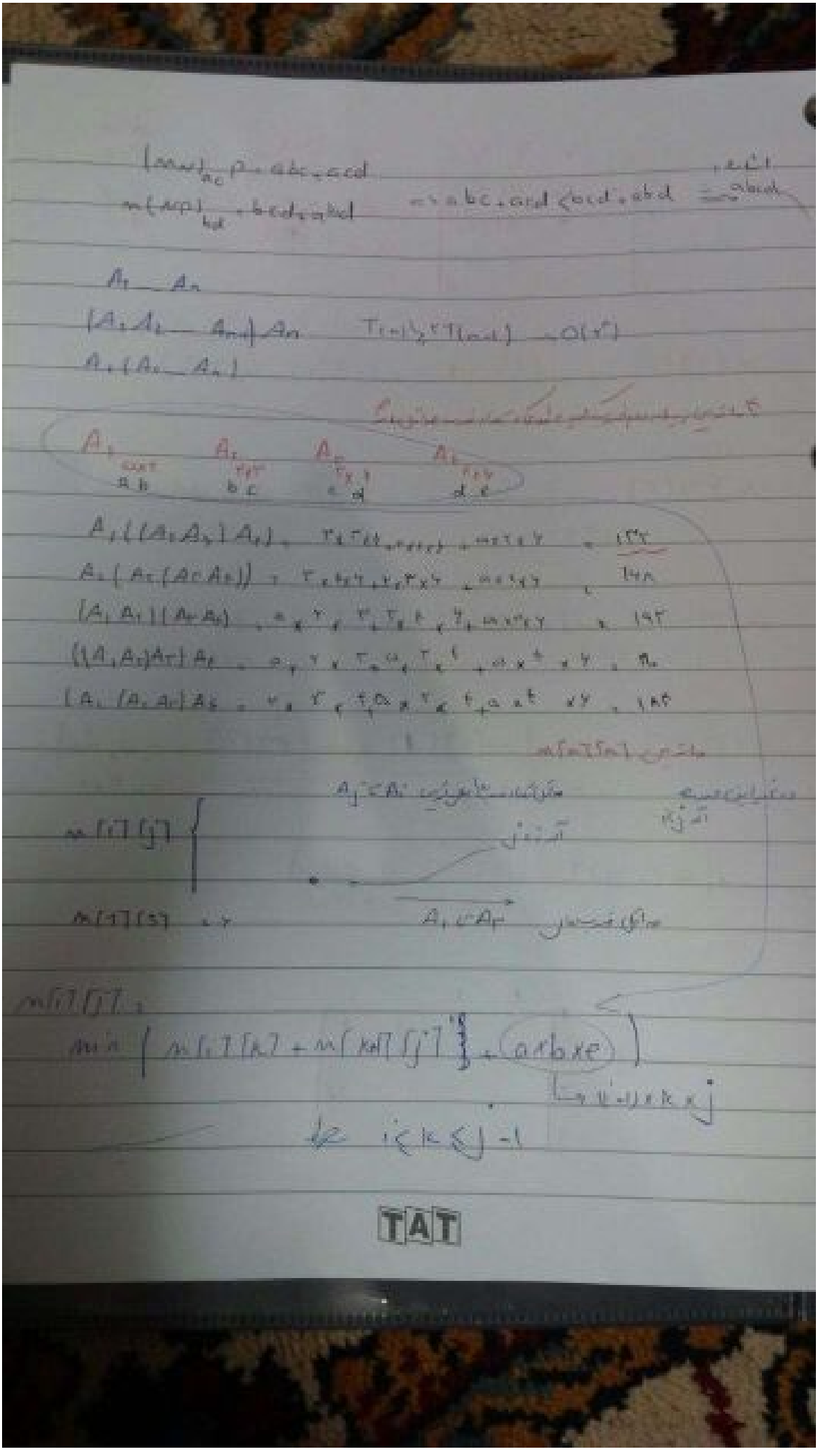
$D[i,j] = \min(D[i,j], D[i,k] + D[k,j])$;

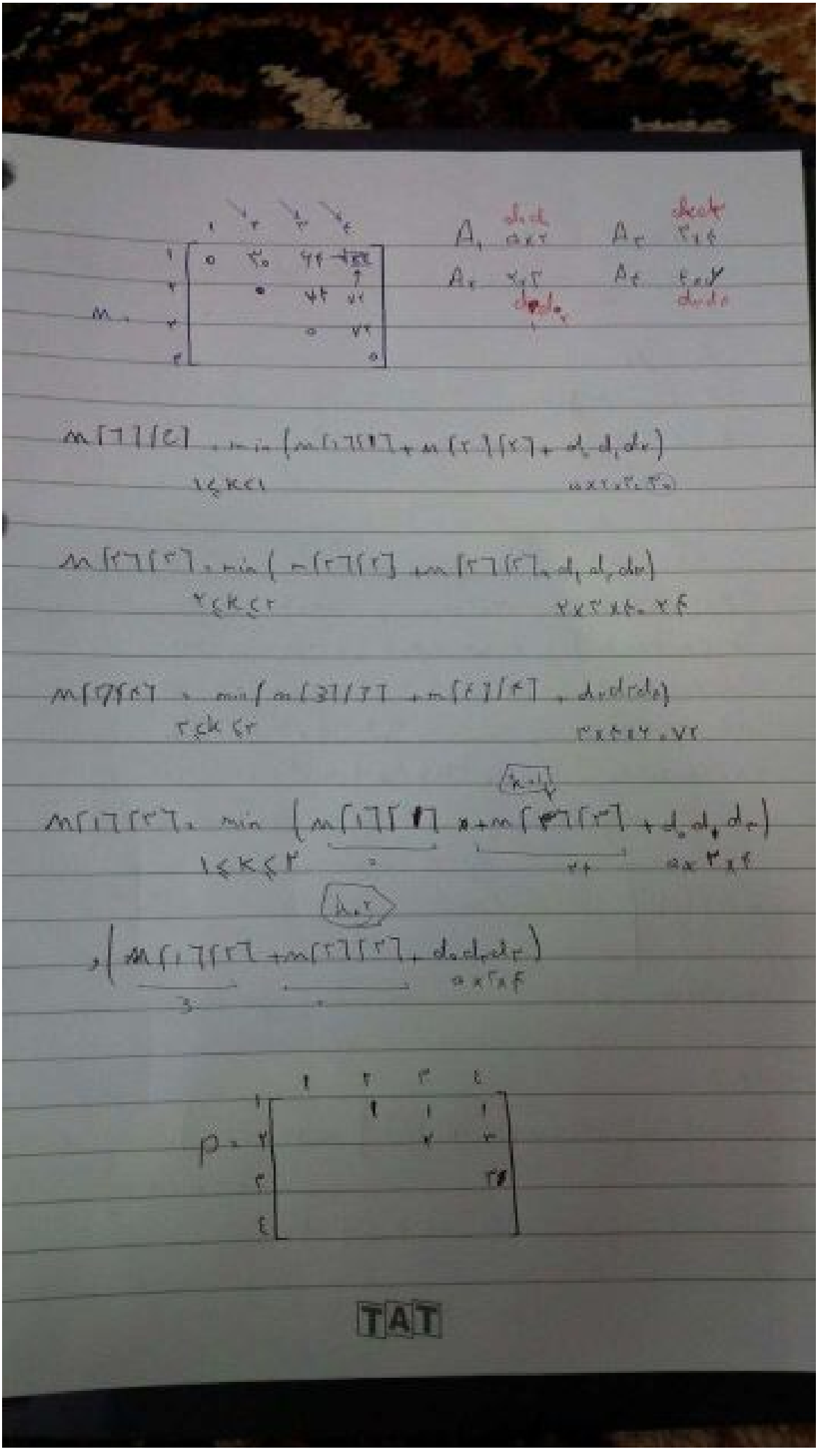
TAT

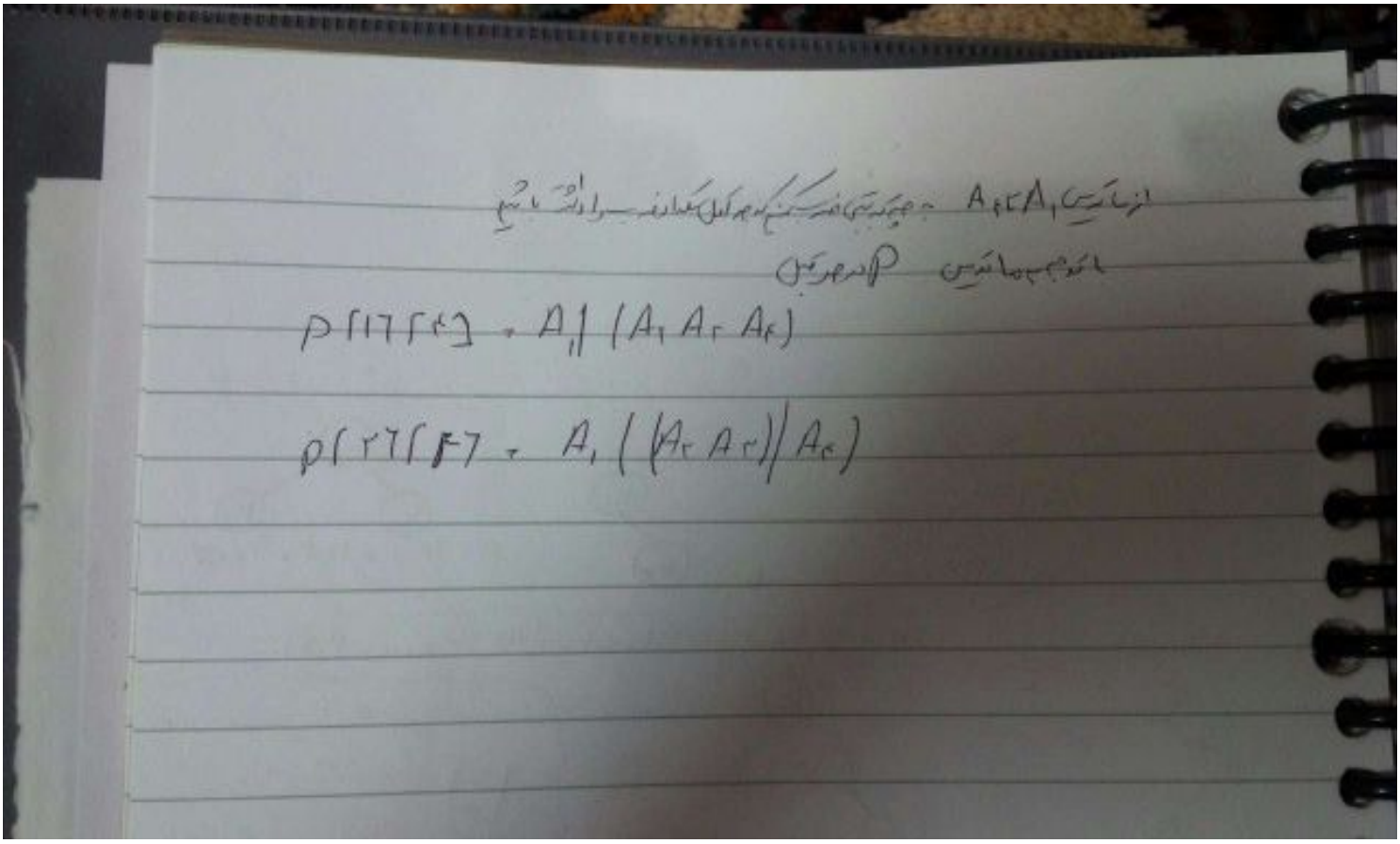
Scanned by CamScanner



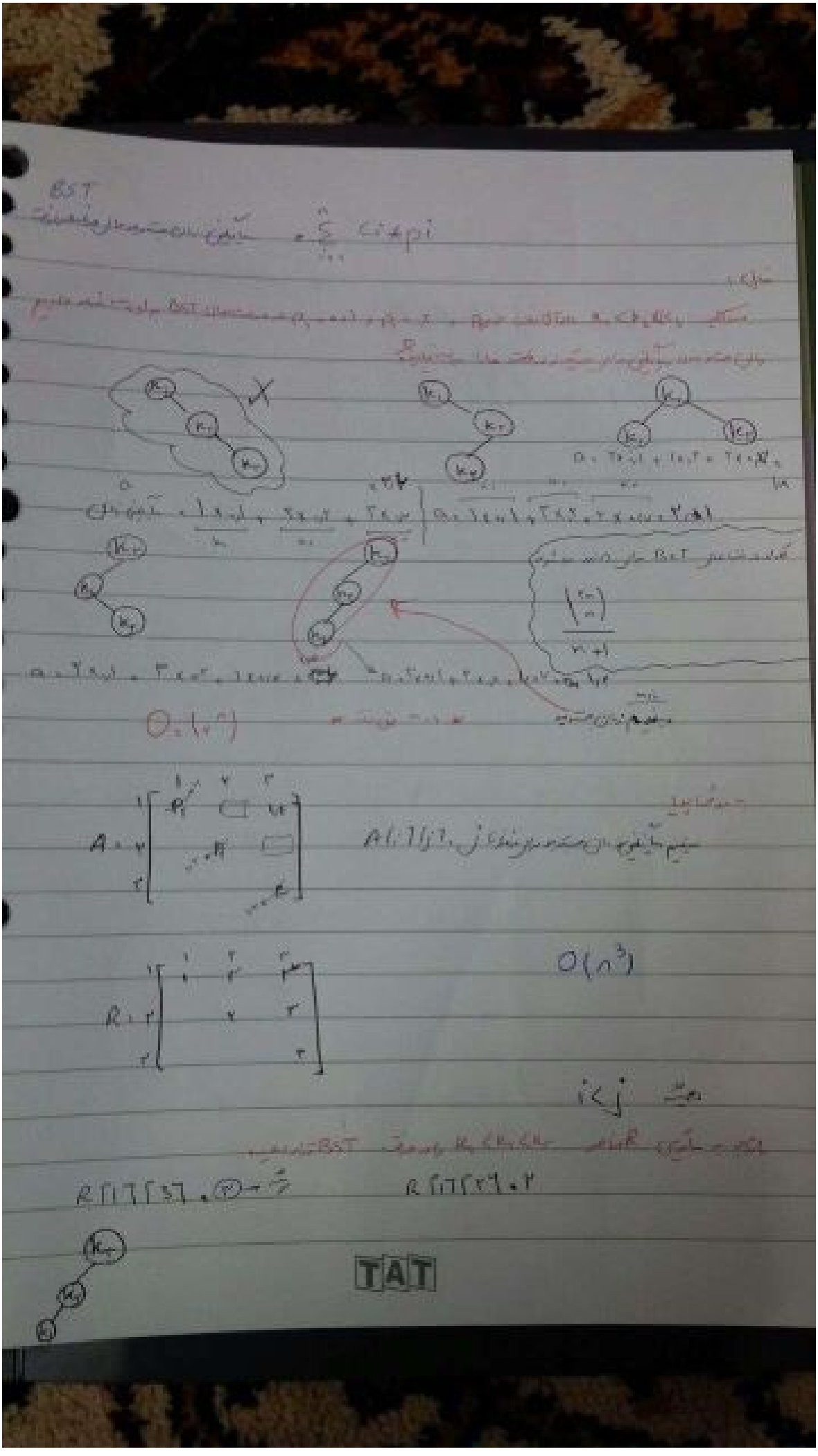




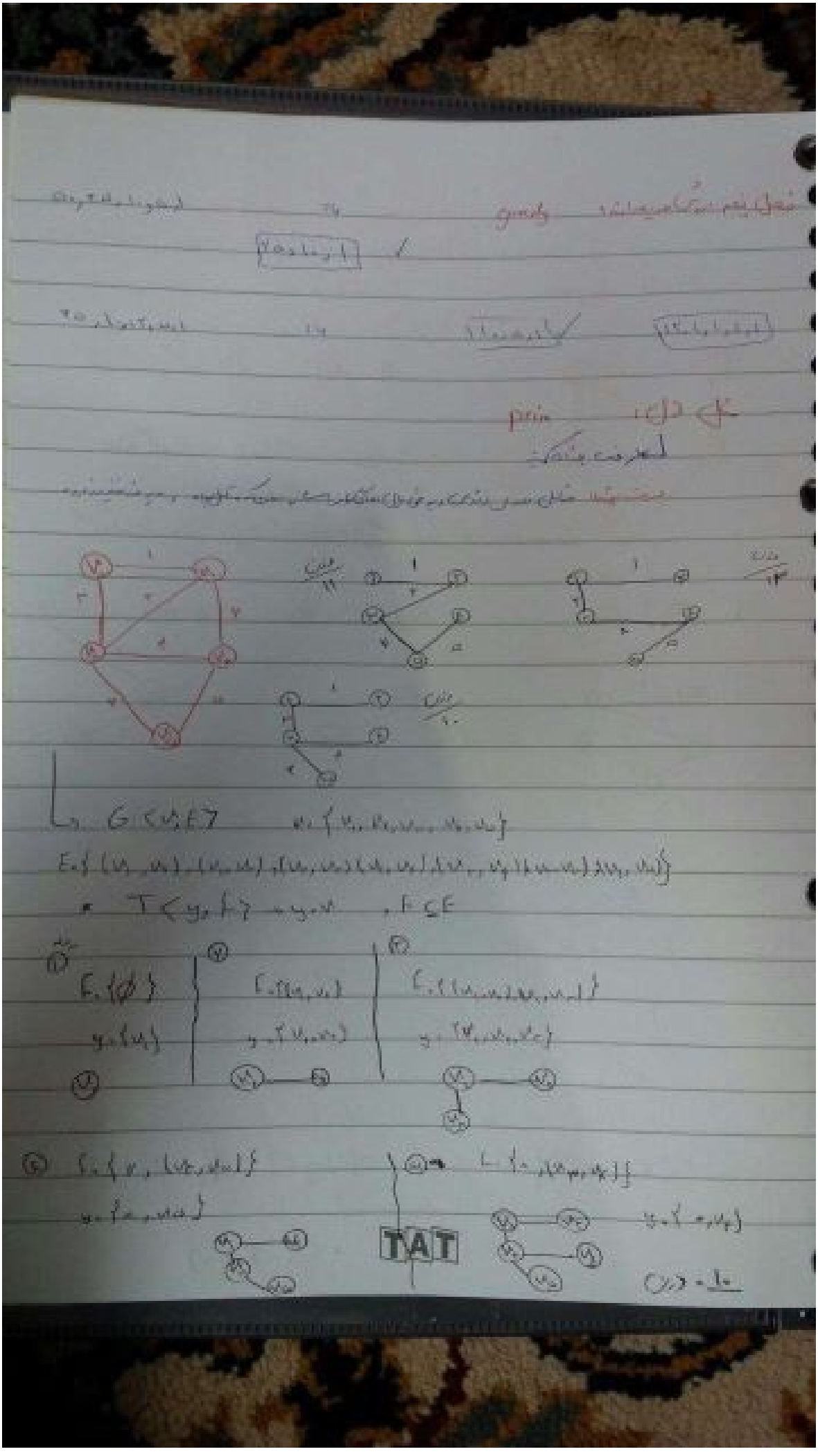




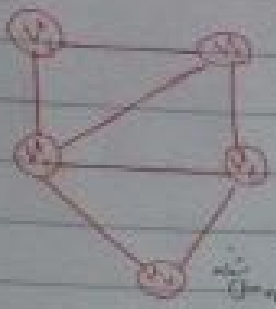
Scanned by CamScanner



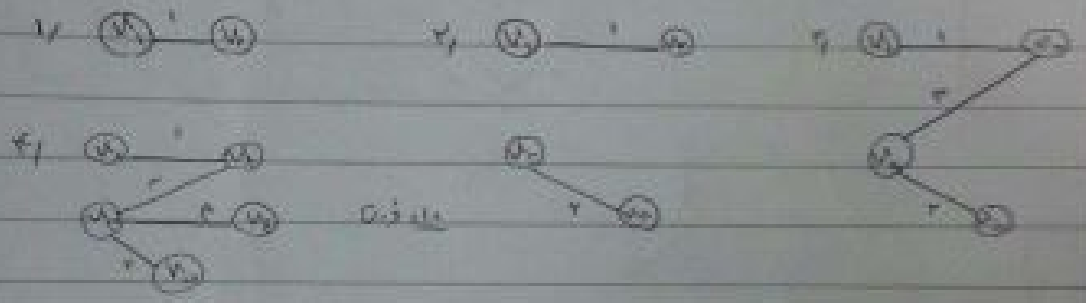
Scanned by CamScanner



Scanned by CamScanner



Kruskal



$O(e \log e)$

$\frac{n(n-1)}{2}$

$n-1 \le e \le \frac{n(n-1)}{2}$ ⟵ $O(e \log e)$, $O(e \log n)$

TAT $O(n \log n)$

