مفاهیم اولیه

به سوالات زیر پاسخ دهید:

1) مفهوم پیچیدگی زمانی چیست، چه کاربردی دارد و به چه عواملی وابسته است؟

2) آیا سریع‌ترین الگوریتم همیشه بهترین گزینه برای استفاده در یک برنامه است؟

3) علامت‌های مجانبی O، Θ و Ω را توضیح دهید.

4) دلیل استفاده معمول از O برای بیان پیچیدگی زمانی الگوریتم‌ها را بنویسید.

5) فرض کنید که تابع زمان اجرا یک چندجمله‌ای به صورت زیر باشد:

$$f(n) = 72n^2 \log n + 26n^2 + 4n + 1$$

نرخ رشد این تابع رو با علامت مجانبی O بیان کنید و توضیح مختصری دهید که چطور و چرا به O آن رسیدید.

تعداد دفعات اجرای هر خط

تعداد هربار اجرا شدن هر خط از کد های زیر را رو به روی آن بنویسید.

1)-----o

```
int func(int a, int b) {
    int sum;
    int temp = a;
    a = b;
    b = temp;
    sum = a + b;
    return sum + temp;
}
```

2)-----o

```
int func(int n);
int func(int n) {
    // This variable is not used.
    int noUse;
    int a = 10;
    int b;
    int temp = 1;
    int result;
    b = 2;
    for (int i = 0; i < n; i++) {
        a += b;
        temp = temp * i;
        b++;
    }
    for (int i = 1; i < n - 1; i *= 2) {
        temp++;
    }
    temp += a;
    result = temp - b;
```

```
    return result;
}
func(10);
```

3)-----o

```
int func(int a, int n) {
    a++;
    if (n <= 0)
        return a;
    else
        return func(a, n-1);
}
```

4)-----o

```
void func(int n, int m, char id) {
    int enemyPower, energy, health = 100;
    if (id == 'k') {
        health += 20;
        energy = 50;
        enemyPower = 8;
    }
    else {
        energy = 25;
        enemyPower = 5;
    }
    for (int i = 0; i < n; i++) {
        health = health + (energy - enemyPower);
        for (int j = m; j > 0; j--) {
            energy--;
            if (energy > 8)
                health += 5;
            else
                health++;
        }
    }
```

```
    if (health <= 0)
        cout << "x_x" << endl;
    else
        cout << "آهای حرومزاده ها ، من هنوز زندم" << endl;
        //charlz gholamhossein
}
```

---------- 🔴 توجه 🔴 ----------

---

1) لازم نیست مجموع **تعداد هربار اجرا شدن خط ها را بدست آورید** ، **صرفا جلوی** هر خط مشخص کنید که چندبار اجرا می شود.

2) در نوشتن تعداد دفعات اجرا ، وقتی به شرط برخورد کردید ، حالت های برقراری شرط و رد شرط را بررسی کنید.

3) تعداد دفعات اجرای تمامی خط های هر کد رو بنویسید. (به تعداد خطوط هر کد ، به همان تعداد پاسخ نیاز است.)

نمونه سوال :

```
void func() {
    int a = 2;
    int b = 1;
    if (a > 1) {
        a = 1;
        b++;
    }
    else {
        b--;
    }
}
```

نمونه پاسخ :

L1) 0

L2) 1

L3) 1

L4) 1

L5) a > 1 ? 1 : 0

L6) a > 1 ? 1 : 0

L7) 0

L8) 0

L9) a <= 1 ? 1 : 0

L10) 0

L11) 0

--> معنی پاسخ نوشته شده در خط 5ام این است که در صورت برقراری شرط (a > 1) ، خط 5ام 1 بار اجرا می شود و در غیر این صورت 0 بار.

الگوریتم مناسب (1)

برای هر کدام از ورودی های زیر ، الگوریتم های داده شده را بر حسب عملکرد بهینه مرتب کند.

$$o---inputs---o$$

$$n_1 : 2$$

$$n_2 : 4$$

$$n_3 : 5$$

$$n_4 : 10$$

$$n_5 : 58$$

$$n_6 : 102$$

$$o---algorithms---o$$

$$T_1(n) : 2n + 50$$

$$T_2(n) : n^2 + 10$$

$$T_3(n) : n!$$

$$T_4(n) : n^2 + n + 1$$

$$T_5(n) : 2\sqrt{n}\log n + 2^n$$

---------- 🔴 توجه 🔴 ----------

---

1) الگوریتم ها را بر حسب عملکرد بهینه نسبت به ورودی داده شده مرتب کنید. یعنی الگوریتمی که سریعتر خروجی را می دهد در مقایسه بزرگتر قرار می گیرد. (سمت راست تر)

2) فقط نوشتن ترتیب نهایی هر ورودی کافی است.

نمونه پاسخ:

- $n : 23 => T_3 < T_5 < T_4 < T_2 < T_1$

مقایسه توابع

نرخ رشد توابع زیر را دو به دو با یکدیگر مقایسه کنید.

(با ذکر راه حل ریاضی، در هر مقایسه، تابعی که نرخ رشد بیشتری دارد را مشخص کنید)

$$3n \quad , \quad c$$

$$3n^2 \quad , \quad 2n^3$$

$$n^2 \quad , \quad 2^{(n+1)}$$

$$3^{\log_{27} n} \quad , \quad \log_5 n!$$

$$75n^2 + 12n + 3 \quad , \quad 23n^2 + 5$$

---------- 🔴 توجه 🔴 ----------

---

1) ارائه ی راه حل ریاضی **الزامی** هست. از حد در بینهایت استفاده کنید:

$$\lim_{n \to \infty} \frac{f(n)}{g(n)}$$

2) در مقایسه اول ، "c" یک مقدار ثابت است.

الگوریتم مناسب (2)

با توجه به شرایط مسئله ، با ذکر دلیل الگوریتم مناسب را برای استفاده مشخص کنید.

(از هر دو دسته الگوریتم مرتب سازی و جست و جو ، یک الگوریتم را انتخاب کنید)

(دلیل رد الگوریتم ها دیگر هم مختصر بیان کنید)

یک بازی سبک شوتر آنلاین در حال توسعه است. بازی به این صورت است که بازیکنان با یگدیگر رقابت کنند و با کشتن بازیکنان دیگر و ... امتیاز خود را بالا ببرند. بازیکنان فقط 5 بار می توانند کشته شوند و بعد از آن دیگر تا پایان بازی نمی توانند بازی کنند و در نتیجه امتیازشان تا پایان بازی ثابت می ماند. در بازی یک نشان قدرت وجود دارد که بازیکنان می توانند آن را در جایی از نقشه بازی یا با کشتن کسی که آن نشان را بدست آورده ، صاحب آن نشان شوند (تا زمانی که کشته نشوند). مزیتی که این نشان دارد این است که امتیاز کسب شده را دوبرابر می کند اما چالشی که ایجاد می کند این است که جایگاه بازیکن در نقشه را مشخص می کند.

می خواهیم به بازی یک جدول رده بندی اضافه کنیم که رتبه بازیکنان را در هر لحظه بر اساس امتیازی که تا اون لحظه کسب کرده اند، نمایش دهد. همچنین می خواهیم بعد از هربار آپدیت جدول (بعد از هربار مرتب سازی) ، نام بازیکنی که نشان قدرت را در اختیار دارد، پیدا کرده و با رنگ زرد نسبت به بقیه بازیکنان در جدول نمایش داده شود. برای این منظور نیاز به یک الگوریتم برای مرتب سازی رتبه بازیکنان و یک الگوریتم برای پیدا کردن بازیکنی که نشان قدرت را به همراه دارد نیاز داریم.

شرایط :

تعداد داده: خیلی زیاد

زمان پردازش: بهتر است کم باشد، زیرا جدول رده‌بندی باید سریع آپدیت شود چون تأخیر در پردازش، نمایش اشتباه رتبه بازیکنان را به همراه دارد. پس هر چه کمتر ، بهتر

محدودیت حافظه: $O(n)$

حالات ورودی:

1) جو آرام است و تعداد بسیار ناچیزی از بازیکنان امتیازشان تغییر می کند.

2) درگیری های شدید بین بازیکنان وجود دارد و امتیاز تعداد زیادی از بازیکنان تغییر می کند.

. . . . .

سه الگوریتم A و B و C برای مرتب سازی مد نظر است و دو الگوریتم D و E برای جست و جو.

از الگوریتم های A و B و C ، یک الگوریتم برای مرتب سازی و از الگوریتم های D و E ، یک الگوریتم برای جست و جو به عنوان الگوریتم مناسب با شرایط انتخاب کنید.

1. الگوریتم های مرتب سازی:

- **الگوریتم A:**

  پیچیدگی زمانی:

  بدترین حالت $O(2^n)$

  بهترین حالت $O(nlogn)$

  پیچیدگی حافظه: O(1)

  پیاده سازی: سخت

  .

- **الگوریتم B:**

  پیچیدگی زمانی:

  بدترین حالت $O(nlogn)$

  بهترین حالت $O(n)$

  پیچیدگی حافظه: $O(n^2)$

  پیاده سازی: آسان

  .

- **الگوریتم C:**

  پیچیدگی زمانی:

  بدترین حالت $O(n^2)$

  بهترین حالت $O(nlogn)$

  پیچیدگی حافظه: $O(logn)$

پیاده سازی: سخت

.

2. الگوریتم های جست و جو:

- **الگوریتم D:**

پیچیدگی زمانی:

داده ها مرتب باشد $O(nlogn)$

در غیر اینصورت $O(logn)$

پیچیدگی حافظه: $O(logn)$

پیاده سازی: سخت

.

- **الگوریتم E:**

پیچیدگی زمانی:

داده ها مرتب باشد $O(n)$

در غیر اینصورت $O(n)$

پیچیدگی حافظه: $O(n)$

پیاده سازی: آسان

پیدا کردن صاحب نشان قدرت با $O(1)$:

- آیا می توانید راه حلی ارائه دهید که بازیکنی که صاحب نشان قدرت است ، موقعیتش در جدول رده بندی بازیکنان با $O(1)$ مشخص شود؟

---------- 🔴 توجه 🔴 ----------

-----

1) تمامی الگوریتم های گفته شده فرضی هستند و به الگوریتم خاصی اشاره نمی کنند.

2) بهترین و بدترین عملکرد الگوریتم ها را با بهترین و بدترین حالتی که مسئله و شرایط آن تامین می کند بررسی کنید.

3) شما در آخر دو الگوریتم را به عنوان الگوریتم مناسب برای توسعه بازی انتخاب می کنید، یک الگوریتم از 3 الگوریتم مرتب سازی مد نظر و یک الگوریتم از 2 الگوریتم جست و جو مد نظر.

4) خواسته اصلی مسئله (که در متن مسئله، پررنگ شده) را با دقت بخوانید.

5) در بخش آخر سوال برای پرسش "پیدا کردن صاحب نشان قدرت با $O(1)$" ، راه حلی ارائه دهید.

پیچیدگی زمانی الگوریتم

پیچیدگی زمانی کد های زیر را با ذکر راه حل بدست آورید.

1)-----o

```
for (int i = 0; i < n; i++) {
    for (int j = 1; j < n; j *= 2) {
        counter++;
    }
}
```

2)-----o

```
for (int i = 0; i <= n; i++) {
    for (int j = 0; j < i; j++) {
        counter++;
    }
}
```

3)-----o

```
for (int i = 2; i <= n+1; i += 2) {
    for (int j = i; j < m*n; j++) {
        counter++;
    }
}
```

4)-----o

```
for (int i = n; i > n/2; i--) {
    for (int j = 10; j < 100; j++) {
        for (int z = 2; z < n; z = pow(z, n)) {
```

```
            counter++;
        }
    }
}
```

5)-----o

```
for (int i = 0; i < n; i++) {
    for (int j = 0; j < pow(2, i); j++) {
        counter++;
    }
}
```

علامت بازی

سوال 1:

اگر

$$f(n) = n^2, \quad g(n) = n^{1+\cos(n)}$$

باشد و n عددی صحیح باشد، کدام یک از عبارات زیر صحیح است؟

۱. f(n) = O(g(n))

۲. f(n) = Ω(g(n))

- هر دو مورد
- فقط مورد 1
- فقط مورد 2
- هیچ کدام از موارد

سوال 2:

فرض کنید

$$f(n) = \Theta(n^2), \quad g(n) = O(n)$$

درستی یا نادرستی عبارات زیر را با اثبات یا مثال نقض مشخص کنید:

۱. g(n) = O(f(n))

۲. g(n) ∉ O(logf(n))

۳. g(n) ∈ ω(f(n))