Dumitrache Adrian - George
324CC

Tema 2 - AA

1)

```
chess ( table )
    while true
        table = choice ({ possible_moves_David })
        ret = f( table )
        if ret == 1
            success
        else if ret == 0 or ret == -1
            fail
        table = choice ({ possible_moves_opponent })
        ret = f( table )
        if ret == 1
            success
        else if ret == 0 or ret == -1
            fail
```

2) a)

```
numerotare_grațioasă_nedet(G)  // G = (V, E)
    nums = ∅
    for each v ∈ V -> O(n)
        num(v) = choice({1, 2, ..., n});
        if num(v) ∈ nums -> O(n)
            fail
        nums = nums ∪ {num(v)}
    nums = ∅
    for each (u, v) ∈ E -> O(m)
        dif = num(u) - num(v)
        if dif ∈ nums -> O(m)
            fail
        nums = nums ∪ {dif}
    for each x ∈ {1, 2, ..., m} -> O(m)
        if x ∉ nums -> O(m)
            fail
    success
```

$O(n^2)$ (first for loop)

$O(m^2)$ (second for loop)

$O(m^2)$ (third for loop)

$\Rightarrow O(n^2 + m^2 + m^2) = O(n^2 + m^2)$

2, b)

```
numerotare_gratioasă_det (G) // G = (V, E)
    permutations [][] = ∅
    generate_permutations(permutations, [1, 2, ..., n], n)

    for each perm ∈ permutations -> O(n!)
        i = 0
        for each v ∈ V -> O(n)
            num(v) = perm[i]
            i++
        nums = ∅ ; OK = 1
        for each (u, v) ∈ E -> O(m)
            dif = num(u) - num(v)
            if dif ∈ nums -> O(m)
                OK = 0
            nums = nums ∪ {dif}
        for each x ∈ {1, 2, ..., m} -> O(m)
            if x ∉ nums -> O(m)
                OK = 0
        if OK
            return 1
    return 0
```

```
generate_permutations ( result [][], perm [], size )
    if size == 1
        result.append ( perm )      } O(1)
        return

    for i = 0 .. size - 1  -> O(n) ... O(2)
        generate_permutations ( result, perm, size - 1 ) -> O(n-1)
                                                            ... O(2)
        if size % 2 == 1
            swap ( perm[0], perm[size - 1] )
        else                                                } O(1)   } O(n!)
            swap ( perm[i], perm[size - 1] )
```

$\Rightarrow O(n!) + O(n! \cdot (n + m^2 + m^2)) =$

$= O(n!(n + m^2))$

3) a)

G_k_to_S_Si_k(G, k) // G=(V, E)

$S = E \to O(m)$

$k = k$

$S_n = \{ \underbrace{\{\}, \{\}, \ldots, \{\}}_{n} \}$

for each $(u, v) \in E \to O(m)$

$S_n[u] = S_n[u] \cup (u, v)$

$S_n[v] = S_n[v] \cup (u, v)$

return (S, S_n, k)

$O(2m) = O(m)$

Algoritmul este:

- determinist (nu se folosește choice)
- tractabil (complexitate polinomială $O(m)$)
- returnează date de intrare pentru Set_Cover pornind de ~~tot~~ la date de intrare pentru k-Acoperire

Obs.:

S = mulțimea muchiilor din graf

$S_n[u]$ = mulțimea muchiilor ce au ca extremitate nodul u

3) c) Demonstrație de echivalență a iesirilor:

Fie $S_n[u] = \{(x, y) \in E \mid x = u$ și / sau $y = u\} =$

= mulțimea muchiilor în care u este extremitate

Fie $S_n[V] = S_n[u_1] \cup S_n[u_2] \cup \ldots \cup S_n[u_{card(V)}]$

$S_n[u] \subseteq E \Rightarrow S_n[V] \subseteq E$

G are o K-Acoperire $\Rightarrow \exists V' \subseteq V$ cu $card(V') = k$

a. î. $\forall (u, v) \in E$ avem $u \in V'$ și / sau $v \in V' \Rightarrow$

$\Rightarrow \forall (u, v) \in E, (u, v) \in S_n[V'] \Rightarrow$

$$\Rightarrow \left.\begin{array}{l} E \subseteq S_n[V'] \\ S_n[V'] \subseteq E \end{array}\right\} \Rightarrow E = S_n[V'] \overset{not.}{\Rightarrow} S = S_n[V'] \Rightarrow$$

$\Rightarrow S_n[u_1] \cup S_n[u_2] \cup \ldots \cup S_n[u_k] = S \Rightarrow$

$\Rightarrow \exists$ k submulțimi date $S_n[u]$ $u \in V$ a. î. reuniunea lor dă ~~sub~~mulțimea $S \Rightarrow$

$\Rightarrow$ Set_Cover$(S, \{S_n[u_1], S_n[u_2], \ldots, S_n[u_n]\}, k) = 1$

$\Rightarrow$ ~~Am~~ demonstrat că dacă G are o K-acoperire, atunci și mulțimea sa de muchii, alături de submulțimile S_n, formează un Set Cover de mărimea k.

Vom demonstra acum și echivalența inversă:

Obs: mulțimile S_n trebuie să ducă la posibilitatea de formare ~~un~~ a unui graf.

Fie o mulțime S și n submulțimi S_n ale sale ~~pentru~~ din care k astfel de submulțimi, odată reunite, formează mulțimea S.

Reprezentăm această mulțime ca un graf în care E=S și V= fiecare nod va fi reprezentat de o submulțime, iar muchiile lor vor fi elementele submulțimilor.

$\exists$ k submulțimi S_n a.î. $S_n[u_1] \cup \ldots \cup S_n[u_k] = S$

Fie V' mulțimea nodurilor reprezentate de cele k mulțimi $\Rightarrow S_n[V'] = S \Rightarrow$

cu card$(V')=k$

~~Stare~~

$\Rightarrow \forall (u,v) \in S$, $(u,v) \in S_n[V']$

$\forall (u,v) \in S$, avem $u \in V'$ și/sau $v \in V'$

Deci: $\exists V' \subseteq V$ cu card$(V') = k$ a.î.

$\forall (u,v) \in E$ avem $u \in V'$ și/sau $v \in V' \Rightarrow$

$\Rightarrow$ G are o k-acoperire

Am demonstrat că:

- $(S, \{S_n[u_1], \ldots, S_n[u_n]\}, k)$ se poate obține din (G, k)
- G are o k-acoperire $\Leftrightarrow$ $S, \{S_n[u_1] \ldots S_n[u_n]\}$ ~~au~~ formează un Set Cover de mărimea k

$\Rightarrow$ k-Acoperire $\leq_p$ Set Cover