теория сложности

> Arora Barak "Complexity Modern Approach" (1st part)

NP-полнота

Характеристики сложности вычисления.
Есть распозователи ($\Sigma^* \to B$) и преобразователи ($\Sigma^* \to \Sigma^*$)

- время: $T(n) = O(f(n))$
- память: $S(n)$
- random: $R(n)$

$DTIME(f) = \{L \mid \exists\, program\ p:$
$1.\ x \in L \implies p(X) = 1, x \notin L \implies p(x) = 0$
$2.\ n = |x| \implies T(p,x) = O(f(n))\}$

$h = (01)^* \in DTIME(n)$

$\widetilde{DTIME}(f) = \{h \mid \dots\}$

палинромы: $Pal \in DTIME_{RAM}(n)$
$Pal \notin DTIME_{TM}(n)$

$P = \cup_{f-polynom} DTIME(f) = \cup_{i=0}^{\infty} DTIME(n^i)$

$p(n)q(n) : p+q, p*q, p(q(n))$
$L_1L_2 \in P : L_1 \cup L_2 \in P, L_1 \cap L_2 \in P, \overline{L_1} \in P, L_1L_2 \in P, L_1^* \in P$

Концепция недетрминированных вычислений

Допускается $\iff$ $\exists$ последовательность переходов, которая приводит к допуску
недетерминирования программа p(x) допускает $\iff$ $\exists$ последовательность недетерминированных выборов, приводящая к допуску
p(x) не допускает $\iff$ $\forall$ последовательности выборов не допуск

def *NTIME(f)* = $\{L \mid \exists$ недетерминированная программа p
$1)\ p(x) - acc \iff x \in L;\ 2)\ T(p,x) = O(f(n))\}$

ex задача о гамильтоновом цикле

```
p(G)
    vis[1..n]: arr of bool
    s = 1
    for i = 1..n
        u = ?{1..n}
        if (vis[u]) return false
        if (su not in EG) return false
        vis[u] = true
        s = u
    if (s != 1) return false
    return true
```

ex `isComposite(z)`, $n = \lceil \log_B z \rceil$, где B - это основание системы счисления

```
a = ?{2..z-1} // T = logn
if z % a = 0  // poly(logn)
    return true
return false
```

Нельзя свопнуть бранчи и сделать проверку на простоту, потому что это `true` и `false` не симметричны в недетерминированных вычислениях (нельзя даже `isPrime(n): return !isComposite(n)`)

---

def *NP* = $\cup_{f-polynome}\ NTIME(f)$, nondeterministic polynomial
stat $P \subset NP$

? $P = NP$

неформально: класс P - класс задач, которые можно решить за полином, класс NP - класс задач, решение которых можно проверить за полином

Σ_1 - класс языков, в которых можно формализовать класс решения, которое можно проверить за полином
$\Sigma_1 = \{L \mid \exists$ полином p, работающая за полином программа R(x, y) - детерминированная
$x \in L \iff \exists\, y$ (называют *сертификат*): $|y| \le p(|x|)\ and\ R(x, y) = 1$
$x \notin L \implies \forall\, y\ (|y| \le p(|x|))\ R(x, y) = 0\}$

ex гамильтонов цикл $Ham \in \Sigma_1$

```
R(G, y):
    y as arr[1..n] of int
    // we can add: y = ?arr[i..n] of {1..n} // O(n)
    vis = arr[1..n] of bool
    for i = 1..n
        if (y[i] y[i mod n+1] not in EG) return false
        if vis[y[i]] return false
        vis[y[i]] = true
    return true
```

Th $NP = \Sigma_1$
$L \in NP$, $L \in \Sigma_1$
неформально: NP - определение на языке недетерминированных формат, Σ_1 - определение на языке сертификатов