Забележка (малка грешка от предния път):
maxRectAreaInHist2(A[1..n], n): //с **bold** шрифт е добавена корекцията

```
maxArea←0
s←Stack.Init() //stack of indexes
Нека си разширим A[0..n+1] = 0 + A[1..n] + 0
for i←0 to n+1
    while s.isEmpty()=FALSE and A[i]≤A[s.top()] do
        idx←s.pop()
        if s.isEmpty() then
            continue
        area←(i-s.top()-1)*A[idx]
        maxArea←max(maxArea, area)
    if s.isEmpty()=TRUE or A[i]>A[s.top()] then
        s.push(i)
return maxArea
```

алтернативно (по-интуитивно) решение: https://www.youtube.com/watch?v=lsQTYlCiU6c

---

P = {задачи за разпознаване, за които съществува алгоритъм с полиномиална времева сложност при най-лоши входни данни}
NP = {задачи за разпознаване, за които при отговор "да" съществува най-къс серфтификат (примерен вход) и съществува алгоритъм с полиномиална времева сложност при най-лоши входни данни, който проверява отговора "да" с помощта на сертификата}

ЗЗР (задача за разпознаване) - задача с 2 възможни изхода - да или не

Деф ЗЗР A се нарича NP-пълна (NP-complete), когато A е NP и ∀B∈NP, B се свежда (reduces) до A в полиномиално време. Пишем $B \propto_p A$.

Деф ЗЗР A се нарича NP-трудна (NP-hard), когато ∀B∈NP, B се свежда (reduces) до A в полиномиално време.

1) SAT (satisfiablility)
Вход: Конюнктивна нормална форма F
Изход: Удволетворява ли се F за някоя оценка на свободните променливи

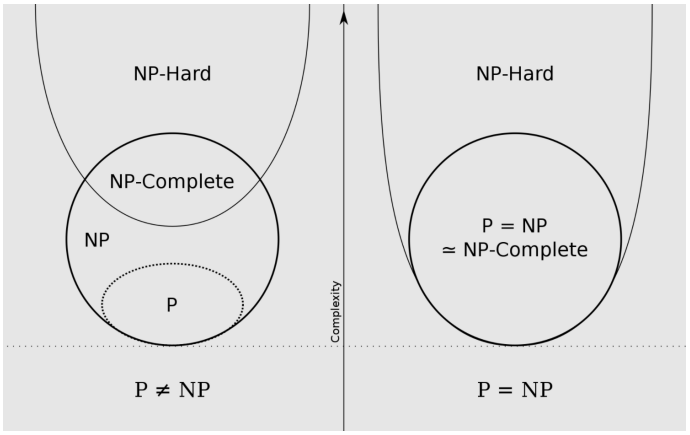
Th Cook-Levin
SAT е NP-пълна (т.е всяка NP задача може да бъде сведена до SAT и съответно SAT е NP)

Тв Една ЗЗР се нарича NP-пълна ↔ те е NP и е NP-трудна

Th Ако ЗЗР C е NP-трудна и $C \propto_p D$, то D е NP-трудна.

2) 3-SAT е NP-пълна (NP & NP-hard)
$(a \vee b \vee c)$ & $(a \vee \neg\, b \vee d)$ & …
Док.
Знаем, че 3-SAT е NP. Сега ще сведем SAT (която е NP-hard) до 3-SAT, откъдето ще получим, че 3-

SAT е NP-hard от където ще имаме, че 3-SAT е NP-complete.
Представяме си че един булев израз е в КНФ, но е съставен само от 1 елементарна дизюнкция. Тоест изглежда нещо от сорта на:
$(a \vee b \vee c \vee d \vee e)$. Сега ще го сведем до 3-SAT:
$((a \vee b \vee t_1) \,\&\, (\neg\, t_1 \vee c \vee t_2) \,\&\, (\neg\, t_2 \vee d \vee e))$
И сега ако си представим, че имаме конюнкция от елементарни дизюнкции, това което може да направим е да махнем скобите след като всеки елементрарен дизюнкт е докаран до 3-SAT.



3) 2-SAT ∈ P
4.1) SubsetSum ∈ NP-complete
4.2) Partiton ∈ NP-complete
5) Задача за раницата
6) Хамилтонов път/цикъл между 2 дадени върха ∈ N-complete
6') Задача за търговския пътник
7) Ойлеров път ∈ P
8) Изоморфен **под**граф ∈ N-complete
9) Изоморфност на графи - това е потенциална задача, която е NP-intermediate
10) Клика ∈ N-complete
11) Антиклика ∈ N-complete
12) Върхово покритие ∈ N-complete
13) Доминиращо множество ∈ N-complete
14) Максимално съчетание ∈ P

Заб Под експоненциално растене имаме предвид по-бързо от полиномиалното.. т.е вкл. $n^{\log(n)}$

Деф MT (машина на Тюринг) работи за време T(n), ако за всеки вход w дължина n, MT прави най-много T(n) прехода

Деф Казваме, че език L е в клас P, ако T(n) - полином: L=L(M), където M е дет. машина на Тюринг, която свърша работа най-късно за T(n) време.

Деф Казваме, че език L е в клас NP, ако T(n) - полином: L=L(M), където M е недет. машина на Тюринг и няма редица о повече от T(n) прехода (т.е всяка дума w∈L / w∉L е разпозната най-много след T(n) прехода.

Заб P е затворено относно операцията допълнение (т.е P≡co-P)

Заб Не се знае дали NP е затворено относно операцията допълнение (очаква се, че не е затворена.. също така се очаква, че L∈NP-C → $L^{\text{compl}} \notin$ NP)

Деф Co-NP (complment) е множеството от всички езици, чиито допълнения са в NP

Пр:
1. USAT (unsatisfiability) - противоречие или невалидни входове
2. TAUT (tautologies) - езика, който съдържа всички тавтологии

Заб Възможно е дори ако P≠NP, да се случи, че NP≡co-NP.

Заб PS някъде се пише PSPACE

Complexity class	Model of computation	Resource constraint
Deterministic time		
DTIME($f(n)$)	Deterministic Turing machine	Time O($f(n)$)
P	Deterministic Turing machine	Time O(poly(n))
EXPTIME	Deterministic Turing machine	Time O($2^{\text{poly}(n)}$)
Non-deterministic time		
NTIME($f(n)$)	Non-deterministic Turing machine	Time O($f(n)$)
NP	Non-deterministic Turing machine	Time O(poly(n))
NEXPTIME	Non-deterministic Turing machine	Time O($2^{\text{poly}(n)}$)

Complexity class	Model of computation	Resource constraint
Deterministic space		
DSPACE($f(n)$)	Deterministic Turing machine	Space O($f(n)$)
L	Deterministic Turing machine	Space O(log n)
PSPACE	Deterministic Turing machine	Space O(poly(n))
EXPSPACE	Deterministic Turing machine	Space O($2^{\text{poly}(n)}$)
Non-deterministic space		
NSPACE($f(n)$)	Non-deterministic Turing machine	Space O($f(n)$)
NL	Non-deterministic Turing machine	Space O(log n)
NPSPACE	Non-deterministic Turing machine	Space O(poly(n))
NEXPSPACE	Non-deterministic Turing machine	Space O($2^{\text{poly}(n)}$)

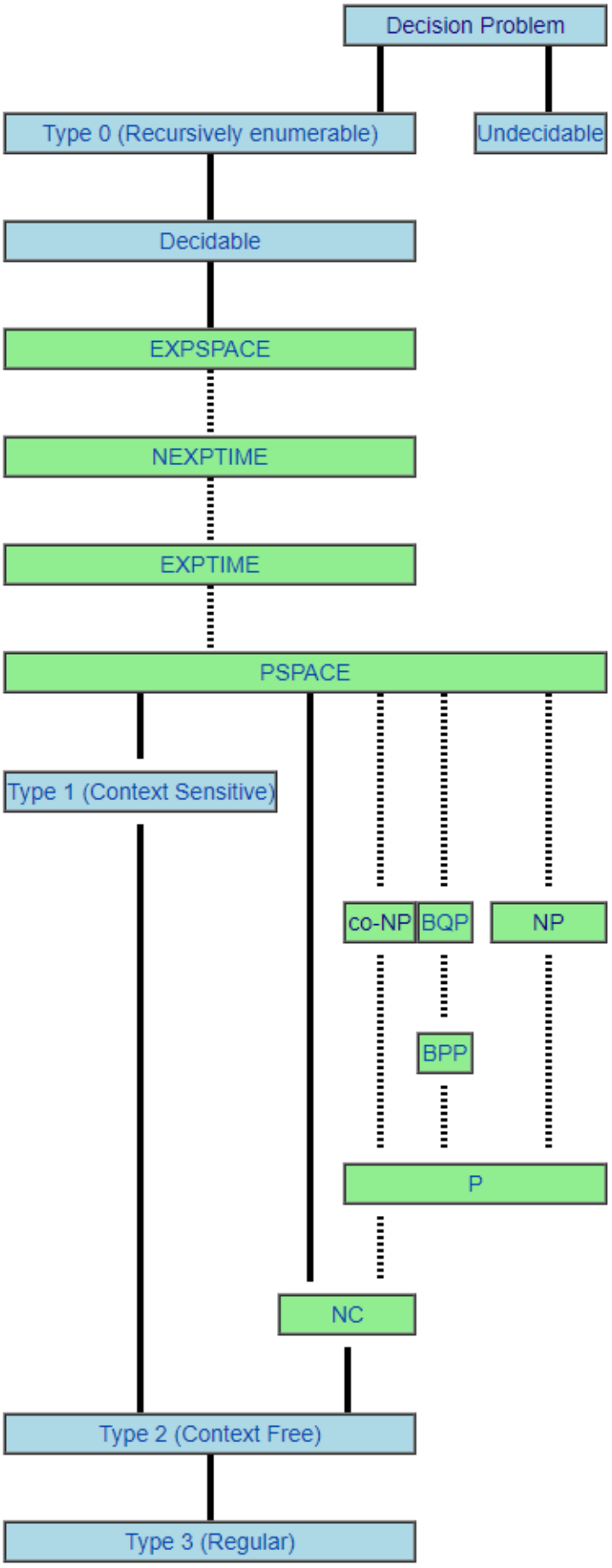
Деф ДМТ M, която за всеки вход w с дължина n, не посещава повече от p(n) клетки от лентата, където p(n) е полином, описва езика L(M). Множеството от всички такива езици ще означаваме с PS.

Деф НМТ M, която за всеки вход w с дължина n, не посещава повече от p(n) клетки от лентата, където p(n) е полином, описва езика L(M). Множеството от всички такива езици ще означаваме с NPS.

Th NPS=PS (PS⊆NPS - очевидна, другата посока е интересната)

Очевидно имаме P⊆PS и NP⊆NPS

От горната теореа и очевидното по-горе имаме: P⊆NP⊆PS=NPS.


Decision Problem
Type 0 (Recursively enumerable)
Undecidable
Decidable
EXPSPACE
NEXPTIME
EXPTIME
PSPACE
Type 1 (Context Sensitive)
co-NP
BQP
NP
BPP
P
NC
Type 2 (Context Free)
Type 3 (Regular)