Граф (4)
Допълнителни задачи

Лекция 13 по СДА, Софтуерно Инженерство
Зимен семестър 2019-2020г
д-р Милен Чечев

План за днес

Ойлеров цикъл в граф

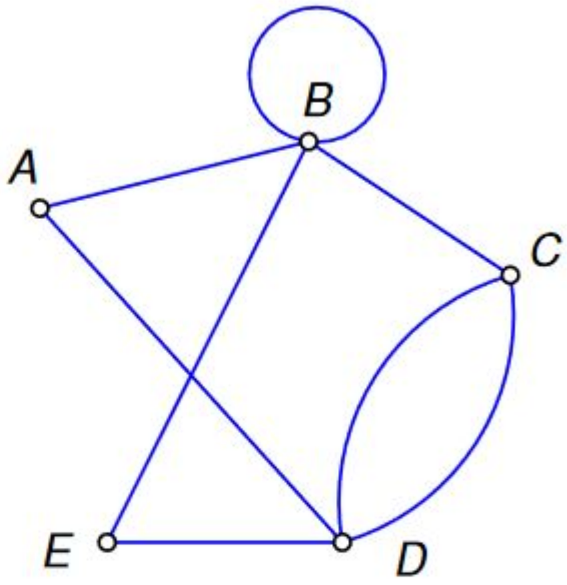
Хамилтонов цикъл в граф

Ойлеров път в граф

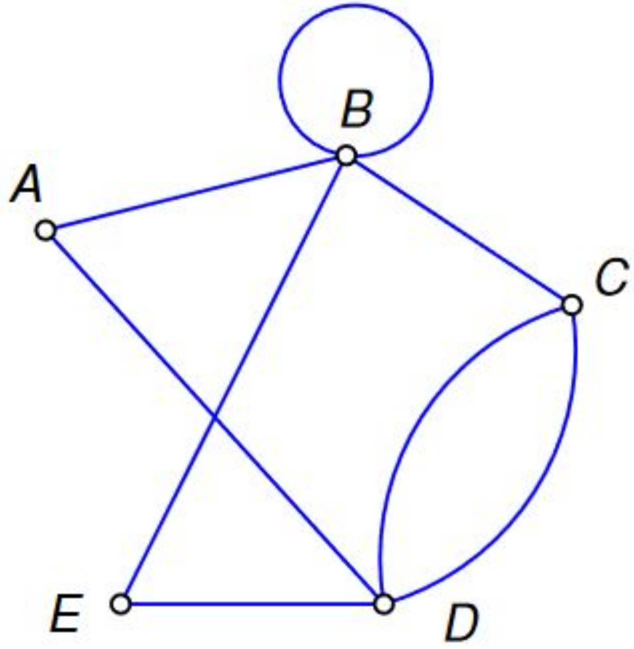
Задача: Да се намери път в граф, който използва всяко ребро в графа точно веднъж.

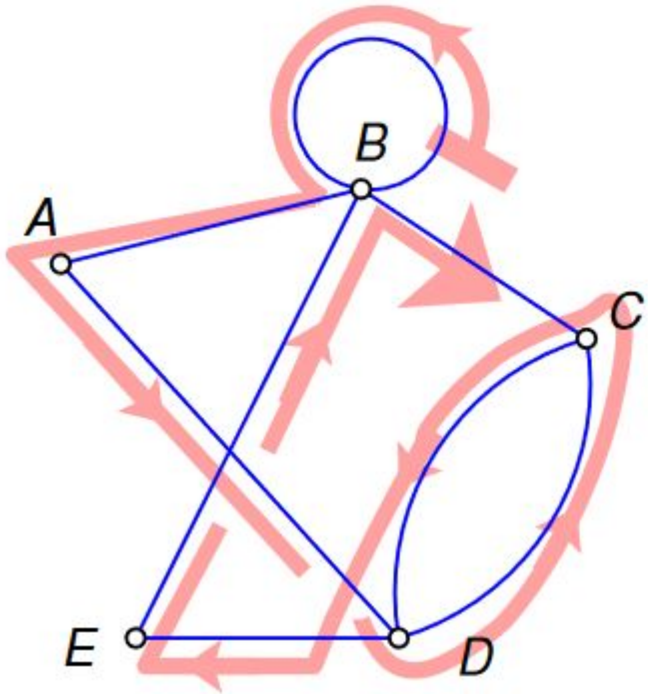
Ойлеров цикъл: Да се намери цикъл в граф, който използва всяко ребро в графа точно веднъж.

Намерете Ойлеров път



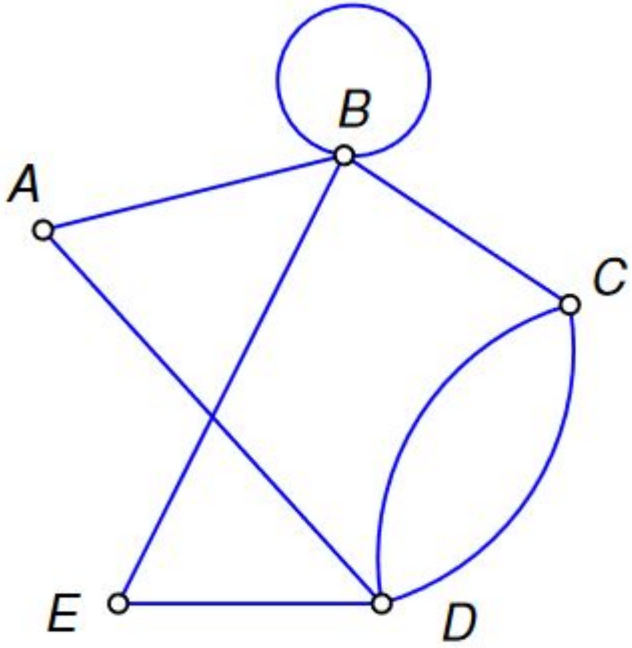
Решение

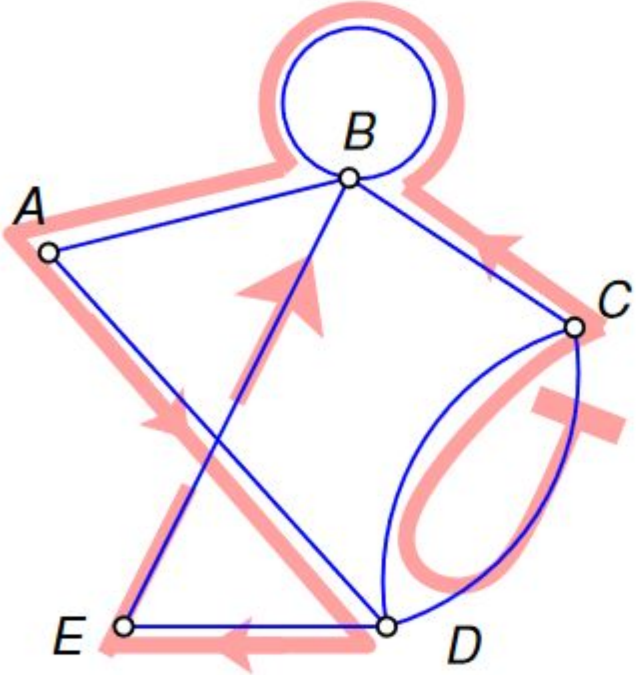




An Euler path: BBADCDEBC

Решение 2

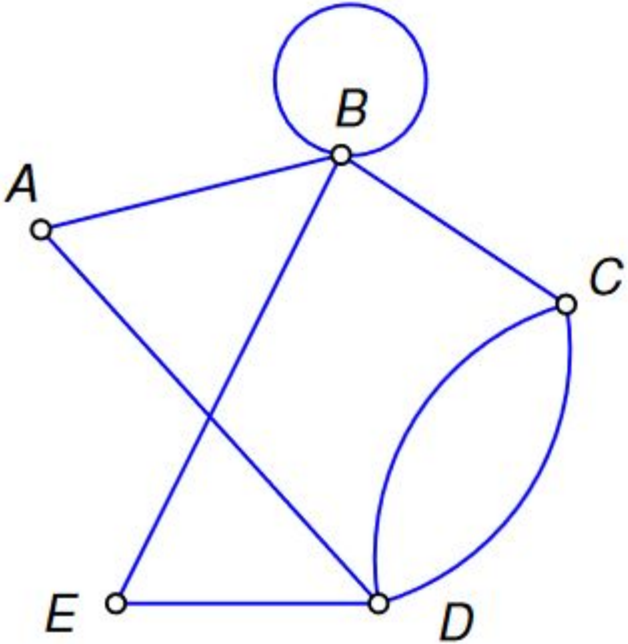


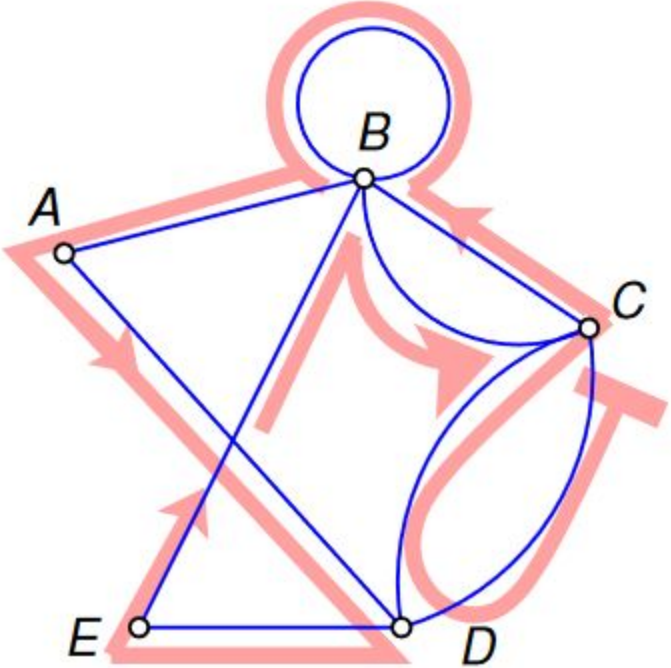


Another Euler path: CDCBBADEB

Ойлеров път -> Ойлеров Цикъл

Добавяме ребро в графа от началото на Ойлеровият път до края.





Проверка за съществуване на Ойлеров цикъл

1. Проверяваме, че графа е свързан
2. Проверяваме броя на ребрата излизащи от всеки граф.Съществува ойлеров път ако точно 2 върха имат нечетен брой ребра.

Имплементация

```
int isEulerian(list<int> *adj){
    if (isConnected() == false)
        return 0;
    int odd = 0;
    for (int i = 0; i < V; i++)
        if (adj[i].size() & 1)
            odd++;
    if (odd > 2)
        return 0;
    return (odd)? 1 : 2;
}
```

Как да намерим Ойлеров път?

Алгоритъм на Флеъри

- Провери дали графа има точно 2 върха с нечетни ребра
- Започни от един от тези два върха и обхождаме графа премахвайки последователно ребра като винаги ако е възможно се избира ребро което няма с изтриването си да прекъсне връзката в графа

Сложност ?

O(E^2)

Как да намерим Ойлеров път?

Алтернатива? Алгоритъм на Хелхолзер...

Обхождаме графа с dfs,като винаги като минем по път го изтриваме. Обхождането започва от единият връх с нечетен брой ребра и ще свърши в другият, като при самото обхождане вероятно част от ребрата няма да бъдат включени. Връщаме се назад и за всеки възел който все още има ребра правим обхождане като вмъкваме намереният цикъл в намереният път до сега.

Сложност?

O(E)

Хамилтонов път в граф

Задача: В ненасочен граф да се отговори дали съществува път, който включва всички възли от графа посещавайки всеки от тях точно един път.

Задачата подобна ли е на Ойлеров път?

Може ли да приложем същият подход?

Решение с пълно изчерпване

1. Генерираме всички пермутации на поредица от върхове
2. Проверяваме дали поредицата от върхове е валиден път в графа

Сложност?

O(V!)

По-добро решение

Backtracking:

Рекурсивно от всеки връх пробваме всеки следващ възможен и т.н. докато се изчерпат.

Сложност:

$O(V!)$

Има ли по-добро решение!

За момента няма открито полиномиално решение(O(N^K)), ако някой го открие ще спечели награда от милион долара ;) : http://www.claymath.org/millennium-problems/rules-millennium-prizes

P->NP->NP Complete->NP Hard

Проблема за Хамилтонов път в граф е NP Complete проблем.

На следващите слайдове ще дефинираме какво е P,NP,NP complete, NP Hard проблеми.

The Class P

A ***decision problem*** is a problem with a yes or no answer.

- Example: Does there exist a pair of duplicates in an array?
- Not a decision problem: How many duplicates are there in an array?

Minor technical point: N is the number of bits needed to specify the input.

We say that a problem is ***in the complexity class P*** if:

- It is a decision problem.
- An answer can be found in $O(N^k)$ time for some k.

Example: Are there two items in an array whose sum is zero? Can solve using technique from discussion (sort, then use two pointers). Runtime is $O(N^2)$.

The Class NP

A ***decision problem*** is a problem with a yes or no answer.

- Example: Does there exist an independent set of size k for graph G?
- Not a decision problem: How big is the biggest independent set for graph G?

We say that a problem is ***in the complexity class NP*** if:

- It is a decision problem.
- A "yes" answer can be <u>verified</u> in $O(N^k)$ time for some k. More precisely, we can verify a specific example of a "yes" answer in $O(N^k)$ time.

Clyde Kruskal has suggested that "VP" is a better name, for "Verifiable in Polynomial Time"

Example: Is there an independent set of size k? Yes, e.g. some set of red vertices Q. To verify, check that all vertices adjacent to vertices in Q are white. Runtime is O(QN), which is $O(N^2)$.

Why Does The Complexity Class P Matter?

Problems in the complexity class P are generally regarded as "easy". For typical N, k, can complete execution within a human lifetime. Example k values:

- Comparison Sorting: 2
- BreadthFirstPaths: 1

Nice features of P:

- $O(N^k)$ is closed under addition and multiplication.
  - Run two P algorithms, overall still in P.
  - Run a P algorithm in N times, still in P.

Exponents for practical problems are typically small.

Why Does The Complexity Classes NP Matter?

Many (most?) practical problems can be cast as a problem in NP:

- Is there a way to route my airplanes at a total cost of less than $1B/yr?
- Is there a way to route the wires inside this microchip with a total path length of less than 1 micrometer?
- Given Z, are there two primes such that X*Y = Z?
- Is there a protein configuration for amino acid sequence X whose total energy is less than Y?

Aside: can generalize idea to problems for which a "no" answer is verifiable.

NP Complete Problems

We'll define a problem π as ***NP-Complete*** if:

- π is a member of NP.
- π *cracks* every other problem in NP.

This raises two questions:

- Are there any NP-Complete problems?
- Do we know how to solve any of them efficiently?

The Cook-Levin Theorem

In 1971, Stephen Cook showed that the 3SAT problem is NP Complete.

- 3SAT is a member of NP.
- 3SAT *cracks* every other problem in NP.

Punchline: If we could efficiently solve 3SAT, we could solve ANY yes/no question whose answer we can efficiently verify.

3SAT: Does there exist a truth value for boolean variables that obeys a set of 3-variable disjunctive constraints: (x1 || x2 || !x3) && (x1 || !x1 || x1)

Levin later (1973) showed a similar result. See Cook-Levin Theorem for more.

Интуитивна дефиниция:

NP проблем: Проблем за който може да проверим дали дадено решение е вярно за полиномиално време.

NP Complete задача:Задача за която ако се намери решение за полиномиално време, то това ще доведе до автоматично намиране на решение за за всички NP проблеми.

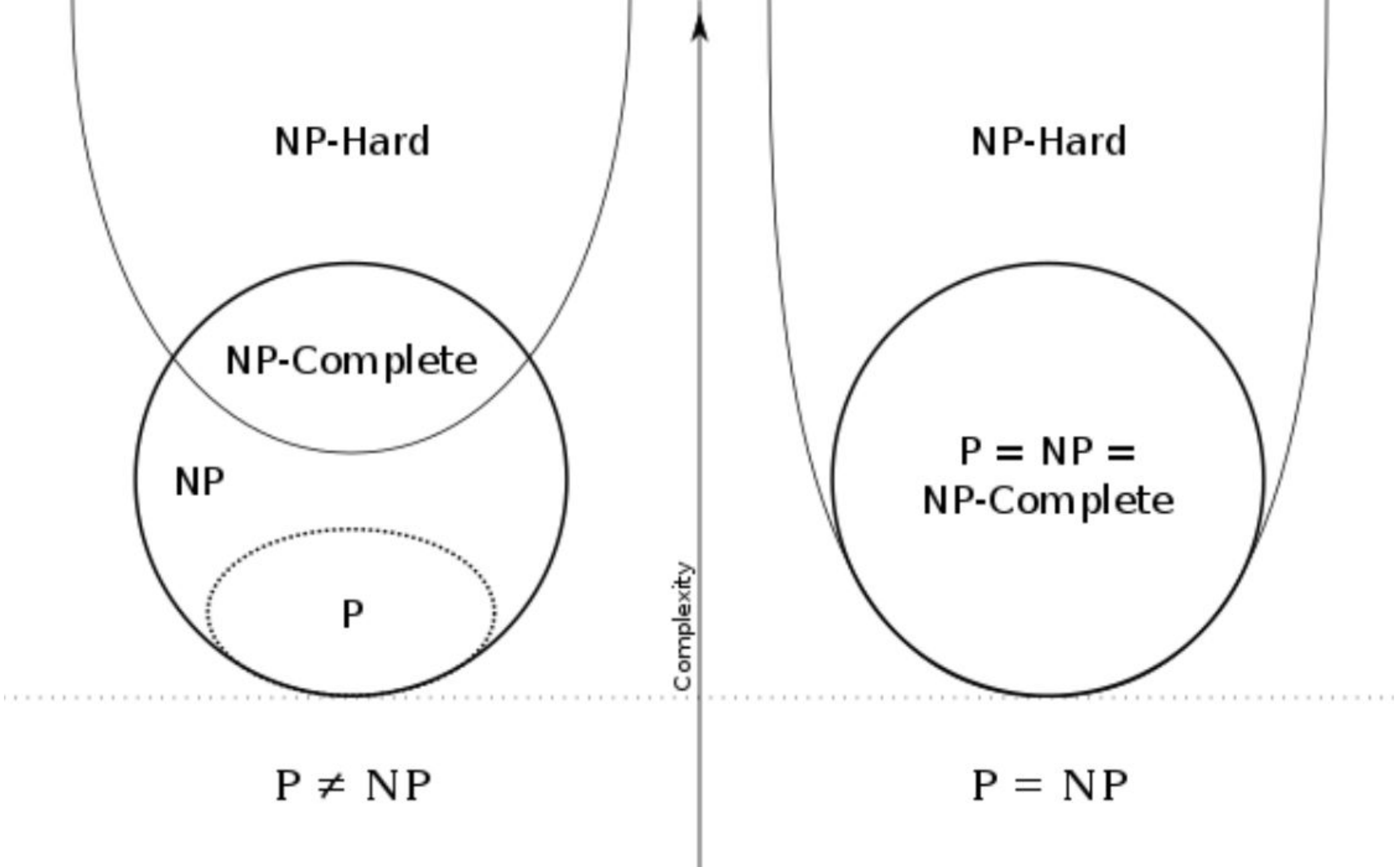
NP Hard проблем

Дефиниция: Проблеми, който са поне толкова трудни колкото най-тежките проблеми в NP (т.е. дори и да се намери полиномиално решение за NP complete проблем, то за тях не означава автоматично, че ще има полиномиално решение за този проблем).

Пример: Задача за пътуващият търговец (Traveling Salesman Problem)

- Да се намери път, който обхожда всички върхове точно 1 път и е с минимална дължина.

За тази задача дори и да ни се върне най-кратък път от програма като решение, не можем да верифицираме резултата за полиномиално време.


NP-Hard
NP-Complete
NP
P
Complexity
NP-Hard
P = NP =
NP-Complete
P ≠ NP
P = NP