数据结构与算法设计

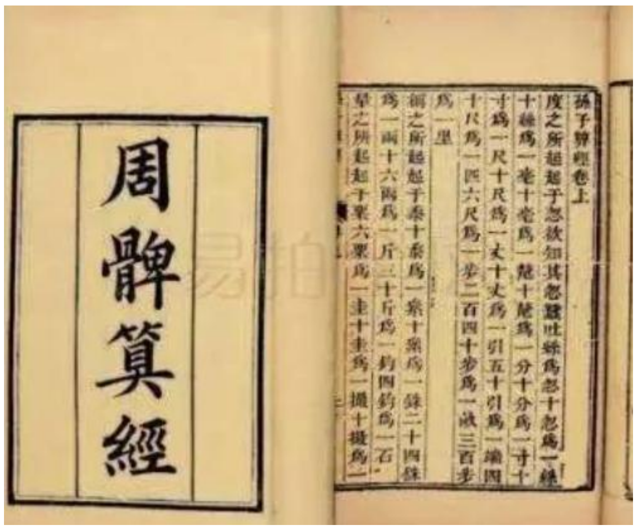
周　可

Mail：zhke@hust.edu.cn

华中科技大学，武汉光电国家研究中心

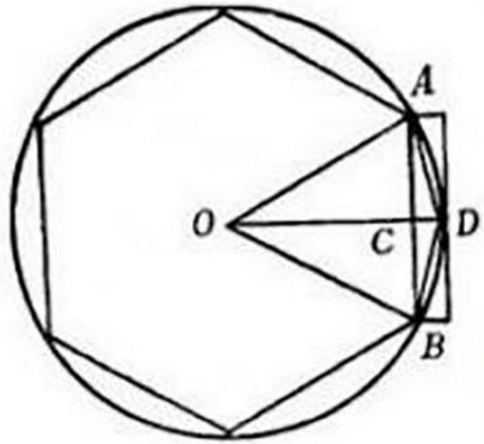
Introduction

周三径一



古人运用近似思想求解问题

刘徽割圆术



割之弥细，所失弥少，割之又割，以至于不可割，则与圆周合体而无所失矣。

Approximation Algorithms

1. Approximation Algorithms

2. The vertex-cover problem

3. The traveling-salesman problem

Background

Many problems of practical significance are NP-complete, yet they are too important to abandon merely. There are three way to get around NP-complete.

1.The actual inputs are small

2.Some important special

3.Near-optimal solutions

<u>Approximation Algorithm</u>

Definition

- ***Optimal solution*** get maximum possible cost or minimum possible cost.

- ***Near-optimal solution*** get the worst cost or the expected cost.

- An algorithm that returns near-optimal solutions is an ***approximation algorithm***.

Q: **how to analyze the performance of approximation algorithm except time or space complexity?**

Approximation ratio

We say that an algorithm for a problem has an ***approximation ratio*** of $\boldsymbol{\rho(n)}$ if for any input of size n, the cost of $\boldsymbol{C}$ of the solution by the algorithm is within a factor of $\boldsymbol{\rho(n)}$ of the cost $\boldsymbol{C^*}$ of an optimal solutions.

$$\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \le \rho(n) \quad C, C^* > 0$$

If an algorithm for a problem has an approximation ratio of $\boldsymbol{\rho(n)}$, we call it a ***$\rho(n)$-approximation algorithm***.

Approximation ratio

- For a maximization problem, $0 \le C \le C^*$;
- For a minimization problem, $0 \le C^* \le C$;
- Approximation ratio is never less than 1.
- A 1-approximation algorithm produces an optimal solution, and an approximation with a large approximation ratio may return a solution that is much worse than optimal.

***Q:* how to get approximation ratio?**

Approximation Algorithms

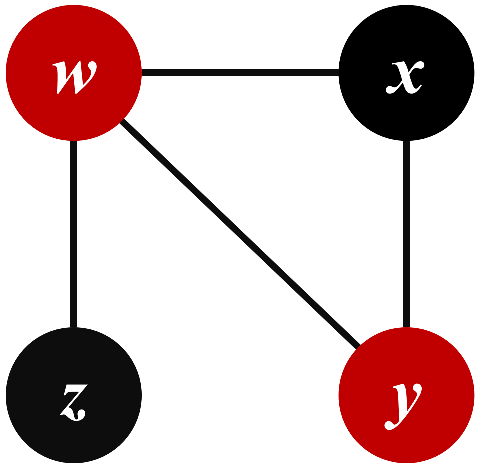
1. Approximation Algorithms

2. The vertex-cover problem

3. The traveling-salesman problem

Background

The vertex cover of *G* is {*w*, *y*}.



It is NP-complete and hard to solve !

Recall that a ***vertex cover*** of an undirected graph $G = (V, E)$ is a subset of $V' \subseteq V$ such that if (u, v) is an edge of G, then either $u \in V'$ or $v \in V'$ (or both). The size of a vertex cover is the number of vertices in it.

The ***vertex-cover problem*** is to find a vertex cover of minimum size in given undirected graph. We call such a vertex cover an ***optimal vertex cover***.

APPROX-VERTEX-COVER

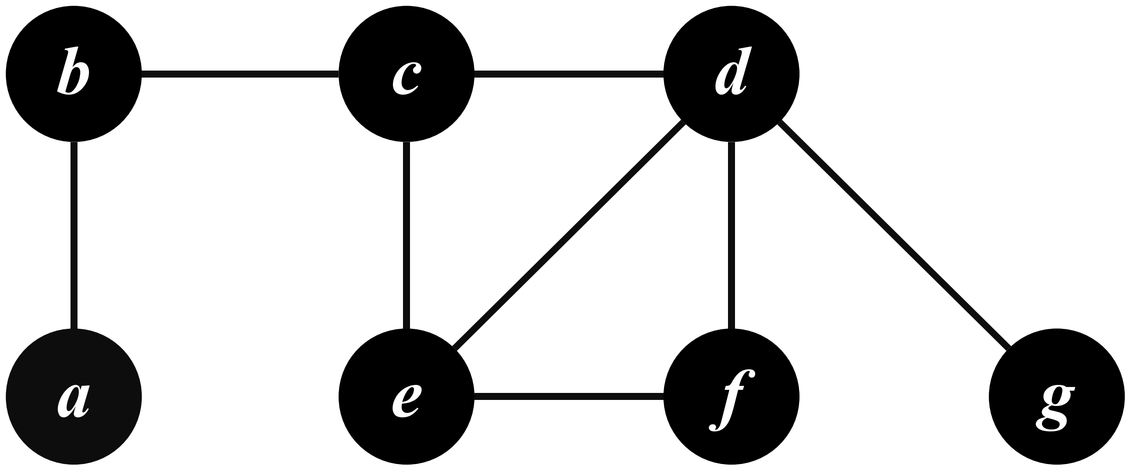
The following approximation algorithm takes as input an undirected graph G and returns a vertex cover whose size is guaranteed to be no more than **twice the size** of an optimal vertex cover.

APPROX-VERTEX-COVER(G)

$C = \emptyset$
$E' = G.E$
while $E' \neq \emptyset$
    let (u, v) be an arbitrary edge of E'
    $C = C \cup \{u, v\}$
    remove from E' every edge incident on either u or v
return C

***Run Time*: O(V + E)**

APPROX-VERTEX-COVER

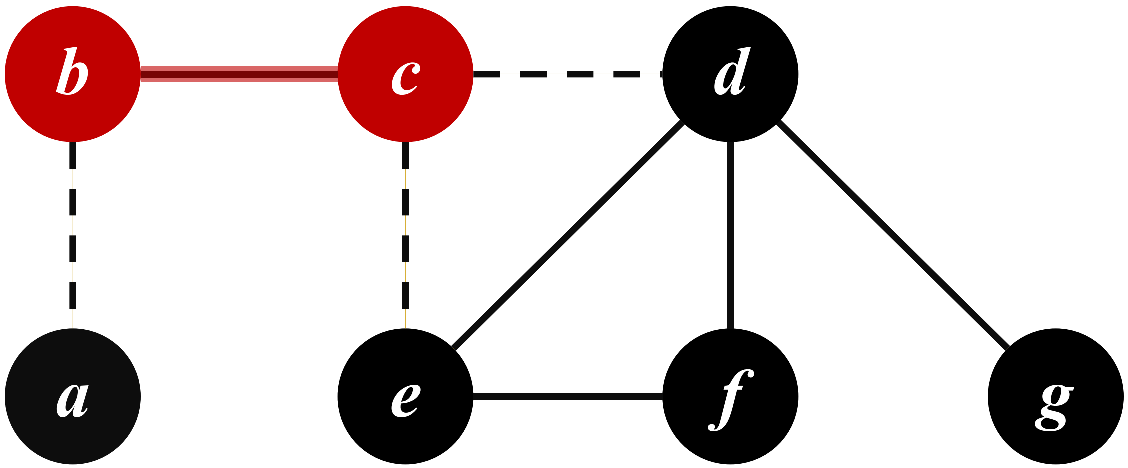


Init

C = {}

E` = {<*a*, *b*>, <*b*, *c*>, <*c*, *d*>, <*c*, *e*>, <*d*, *e*>, <*d* , *f*>, <*d*, *g*>, <*e*, *f*>}

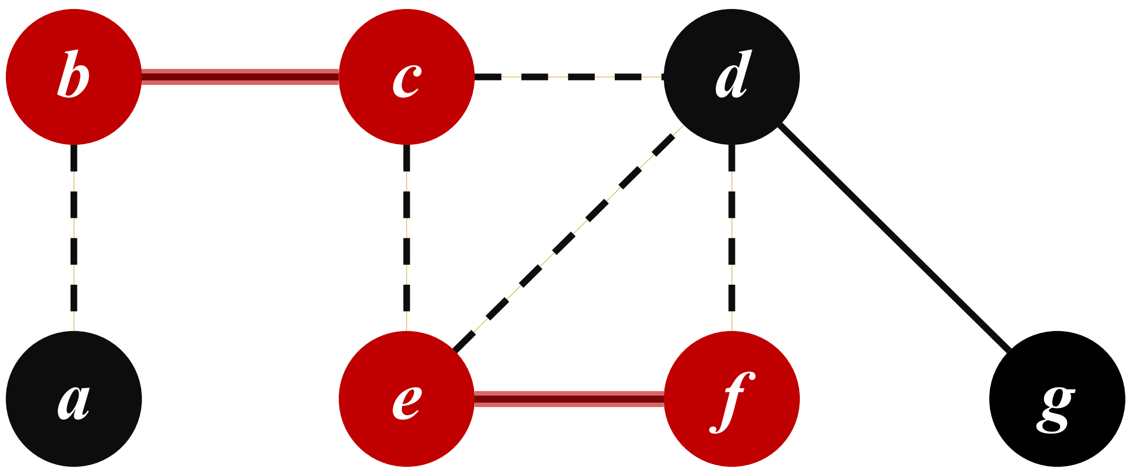
APPROX-VERTEX-COVER



C = {*b*, *c*}

E\` = {<*d*, *e*>, <*d* , *f*>, <*d*, *g*>, <*e*, *f*>}

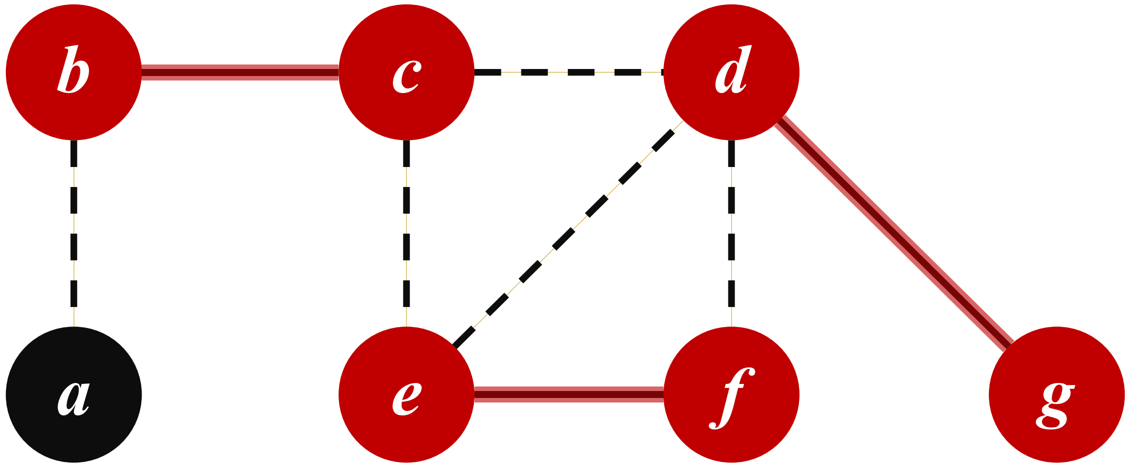
APPROX-VERTEX-COVER



$C = \{b, c, e, f\}$

E\` = {<*d*, *g*>}

APPROX-VERTEX-COVER

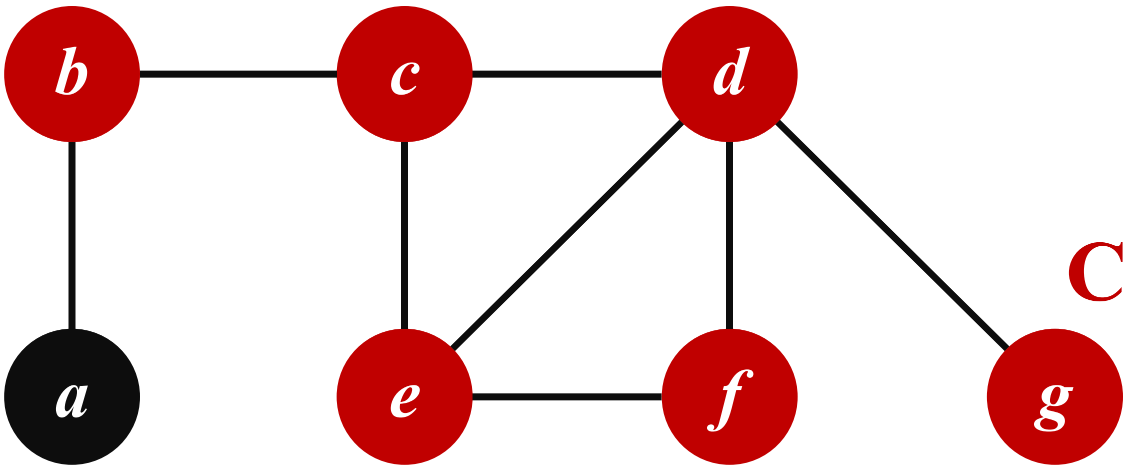


End

$C = \{b, c, e, g, d, g\}$

$E` = \{\}$

APPROX-VERTEX-COVER



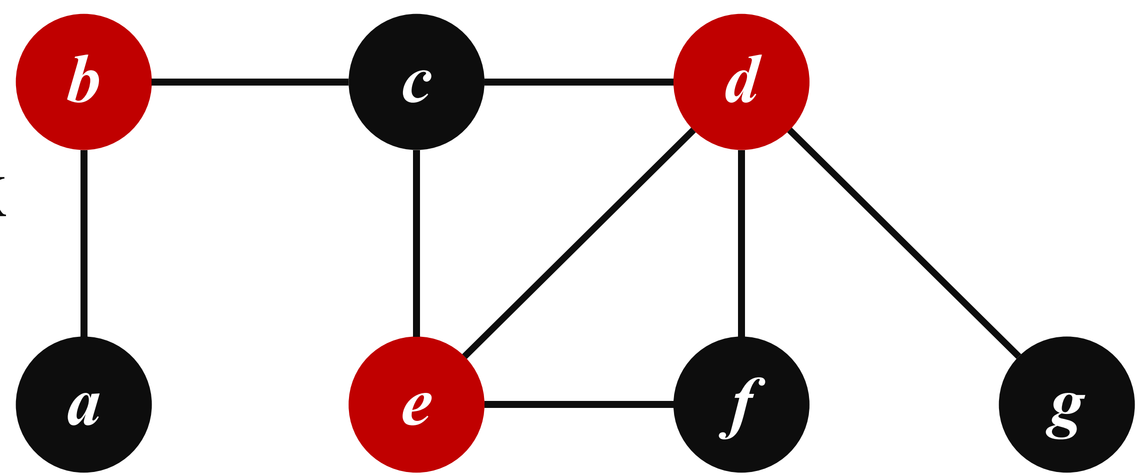
Near-optimal vertex cover

C = {*b*, *c*, *e*, *g*, *d*, *g*}

Approximation ratio=2

Optimal vertex cover

C = {*b*, *e*, *d*}



Proof: *Approximation ratio*

- Let A denote the set of edges that line 4 of APPROX-VERTEX-COVER picked.
- No two edges in A are covered by the same vertex from C^*, and we have ***the lower bound***: $|C^*| \geq |A|$
- Each execution of line 4 picks an edge for which neither of its endpoints is already in C, yielding ***an upper bound***:

$|C| = 2|A|$ **Thus:** $|C| = 2|A| \leq 2|C^*|$

Approximation Algorithms

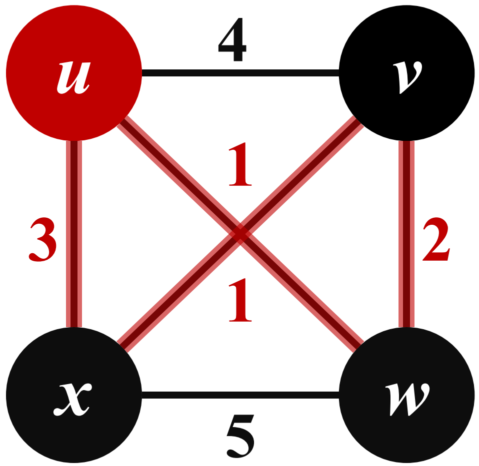
1. Approximation Algorithms

2. The vertex-cover problem

3. The traveling-salesman problem

Background

It is NP-complete problem!



Red edges represent a minimum-cost tour, with cost 7.

Traveling-salesman problem is given a complete undirected graph $G = (V, E)$ that has a nonnegative integer cost $c(u, v)$ associated with each edge $(u, v) \in E$, and must find a Hamiltonian cycle of G with minimum cost.

Let $c(A)$ denote the total cost of the edges in the subset $A \subseteq E$:

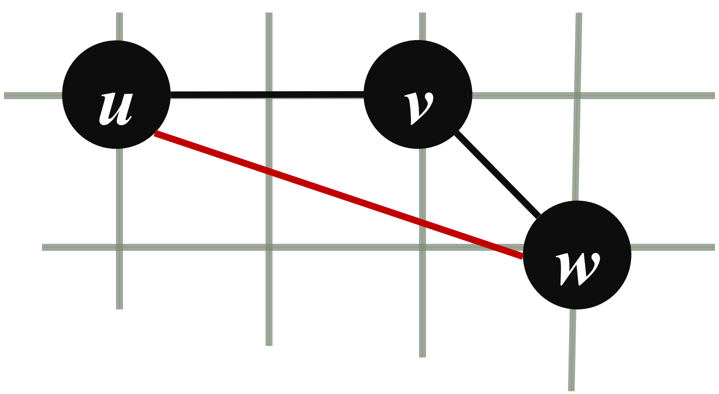
$$c(A) = \sum_{(u,v)\in A} c(u,v)$$

3.1 The traveling-salesman problem with the triangle inequality

The triangle inequality

In many practical situations, the least way to go from a place u to a place w is go directly, with no intermediate steps. **If cutting out an intermediate stop never increase the cost, we say that the cost function c satisfies *the triangle inequality***: if, for all vertices $u, v, w \in V$,

$$c(u,w) \leq c(u,v) + c(v,w)$$



The ordinary euclidean distance cost function satisfies the triangle inequality.

APPROX-TSP-TOUR

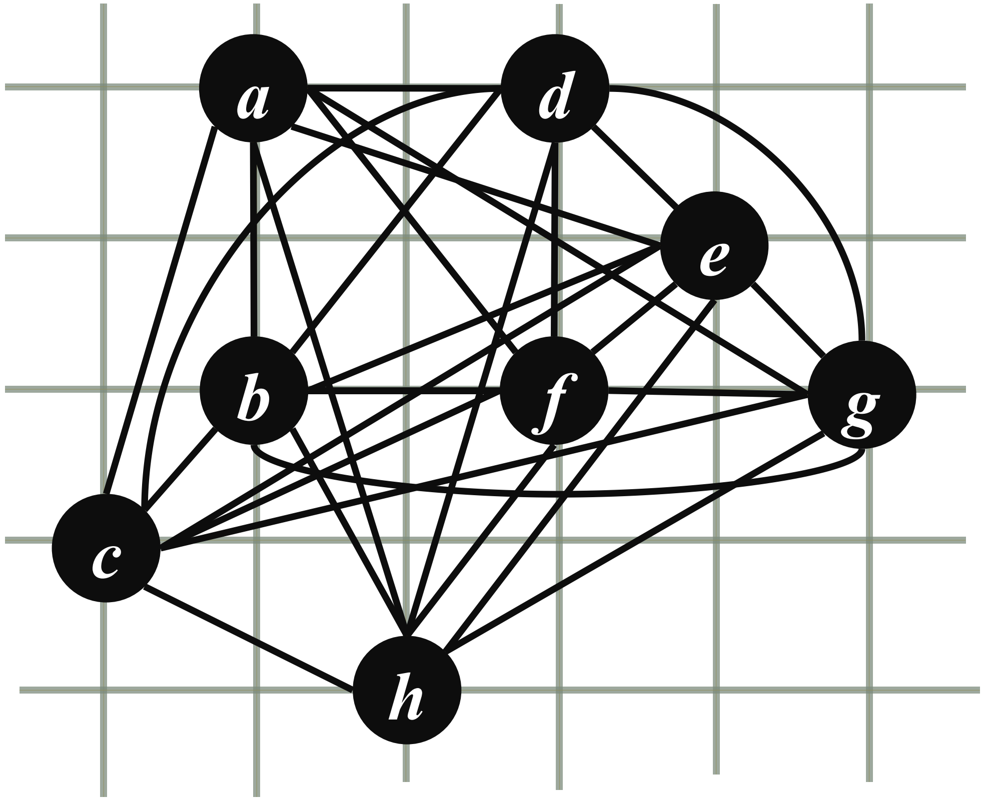
The following algorithm implements **with MST-PRIM**. The parameter G is a complete undirected graph, and the cost function c satisfies ***the triangle inequality***.

APPROX-TSP-TOUR(G, c)

1 select a vertex $r \in G.V$ to be a "root" vertex
2 compute a minimum spanning tree T for G from root r
using MST-PRIM(G, c, r)
3 let H be a list of vertices, ordered according to when they are first visited
in a preorder tree walk of T
4 **return** the hamiltonian cycle H

***Run Time*: O(V^2)**

APPROX-TSP-TOUR

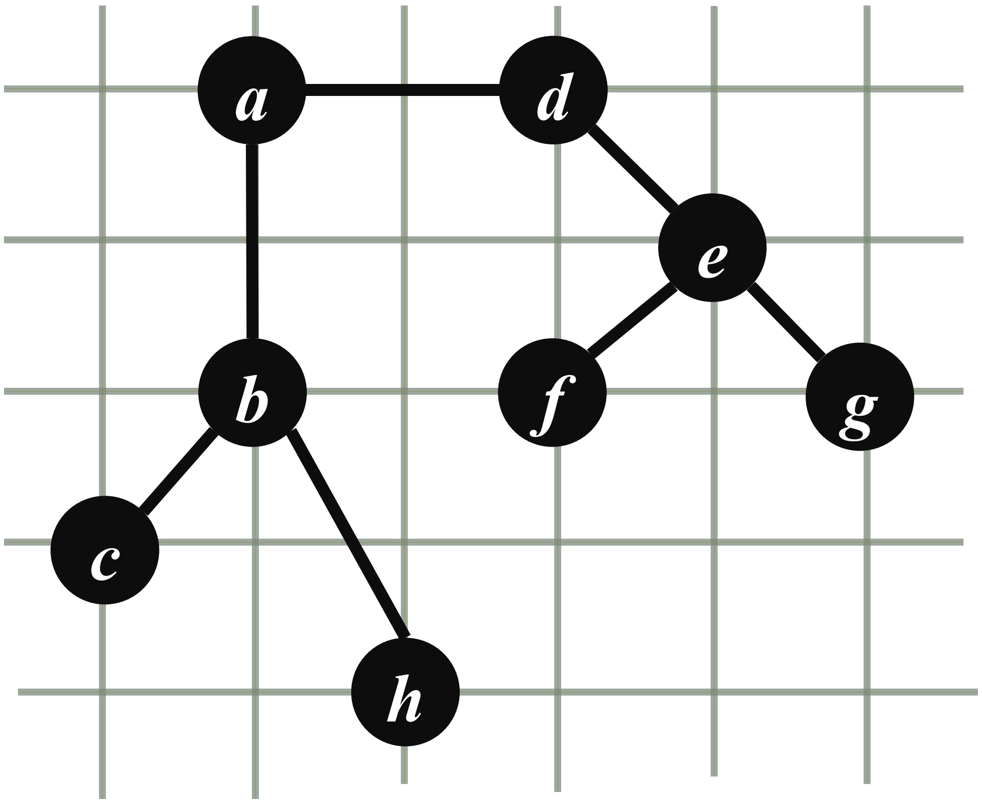


Root vertex = *a*

$c(a, b) = 2$

Cost function c is the ordinary euclidean distance

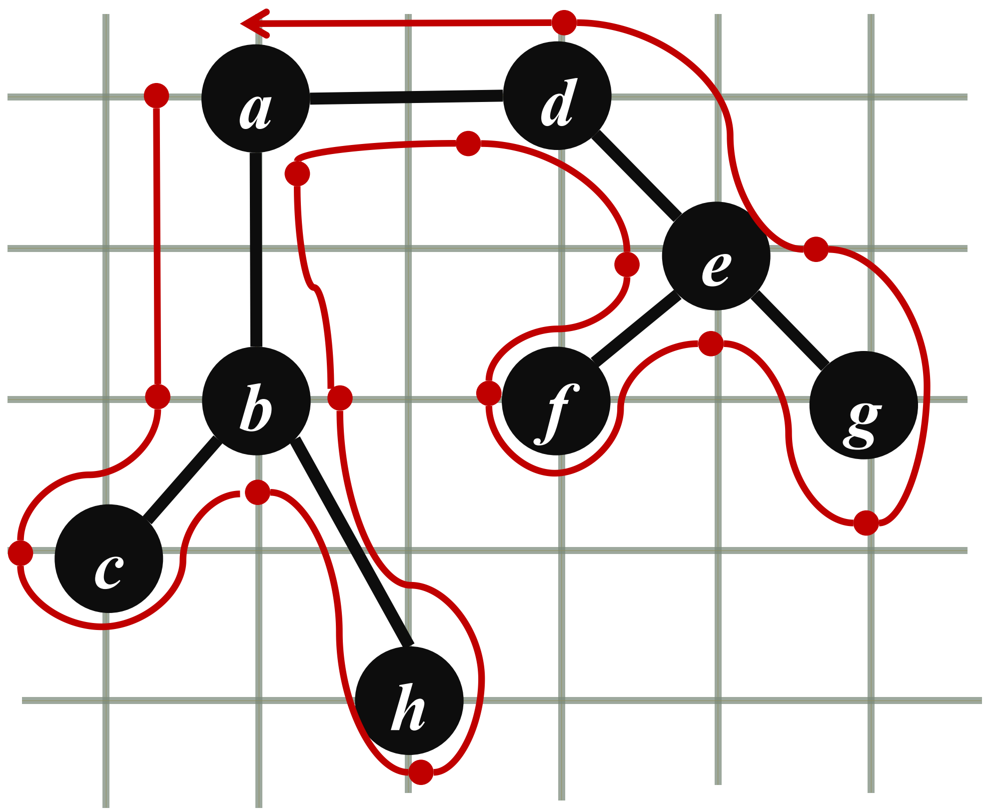
APPROX-TSP-TOUR



Root vertex = *a*

MST-PRIM O(V^2)

APPROX-TSP-TOUR



Preorder tree walk

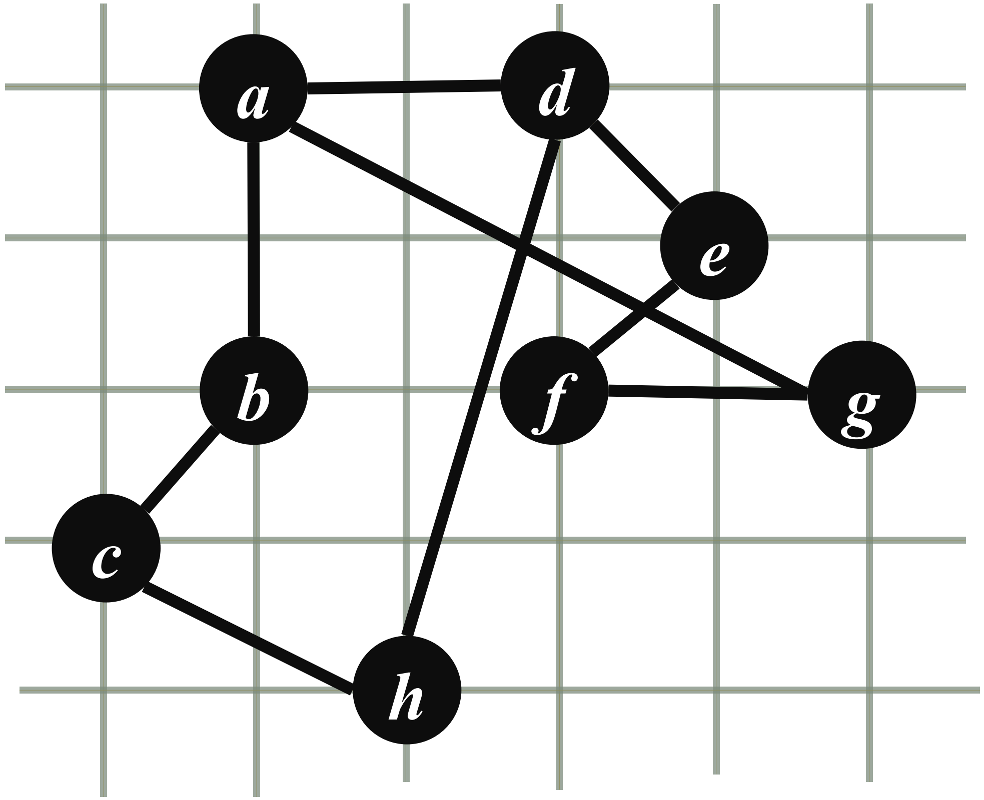
{a, b, c, h, d, e, f, g}

full walk (red line)

{a, b, c, b, h, b,* a*, d, e, f, e, g, e, d,* a*}

A *full walk* of a tree lists the vertices when they first visited and also whenever they are returned to after a visit to a subtree.

APPROX-TSP-TOUR



full walk

$\{a, b, c, b, h, b, \mathrm{a}, d, e, f, e, g, e, d, \mathrm{a}\}$

⇩ *triangle inequality*

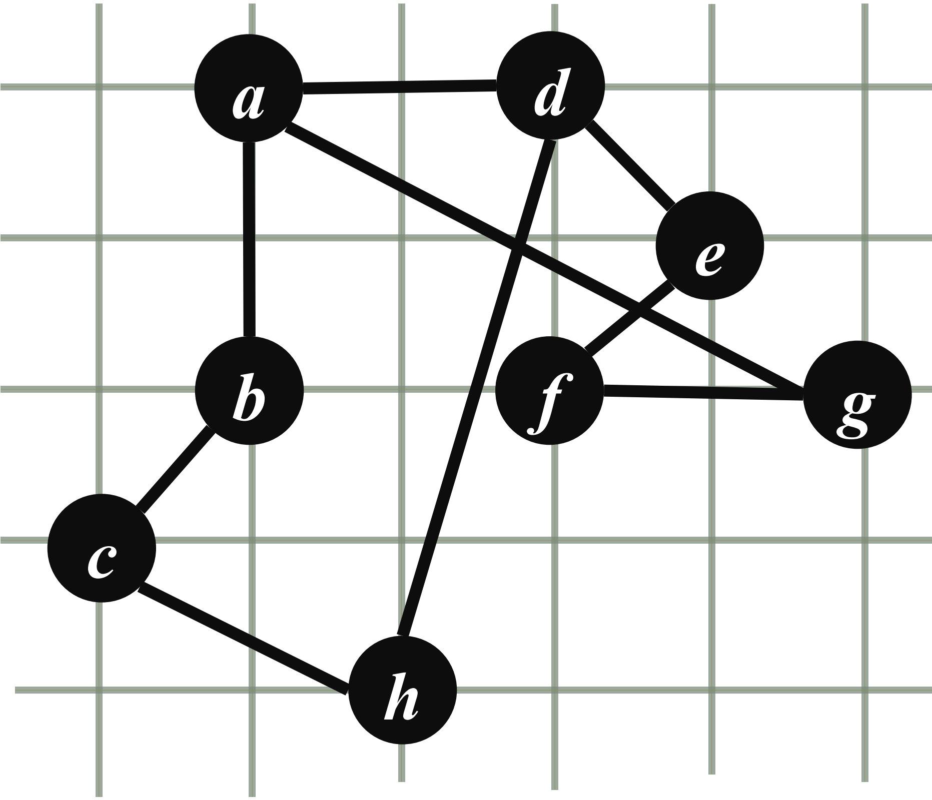
Approx-TSP-tour

$\{a, b, c, h, d, e, f, g, \mathrm{a}\}$

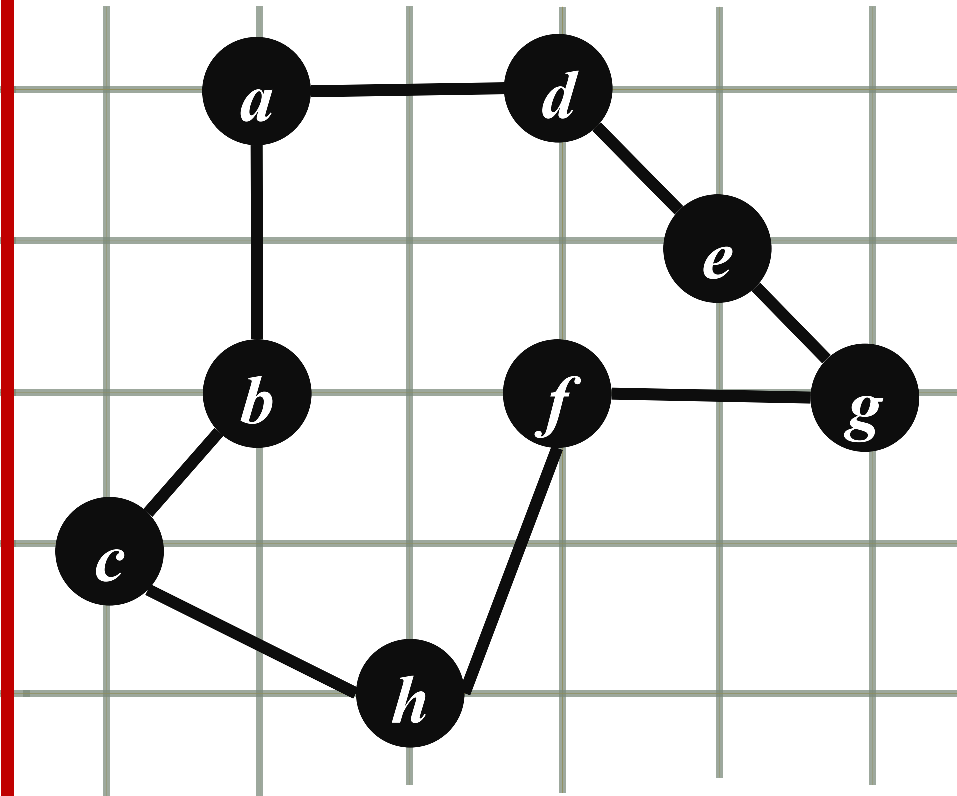
E.g. $c(c, h) \leq c(c, b) + c(b, h)$

$c(g, a) \leq c(g, e) + c(e, d) + c(d, a)$

APPROX-TSP-TOUR



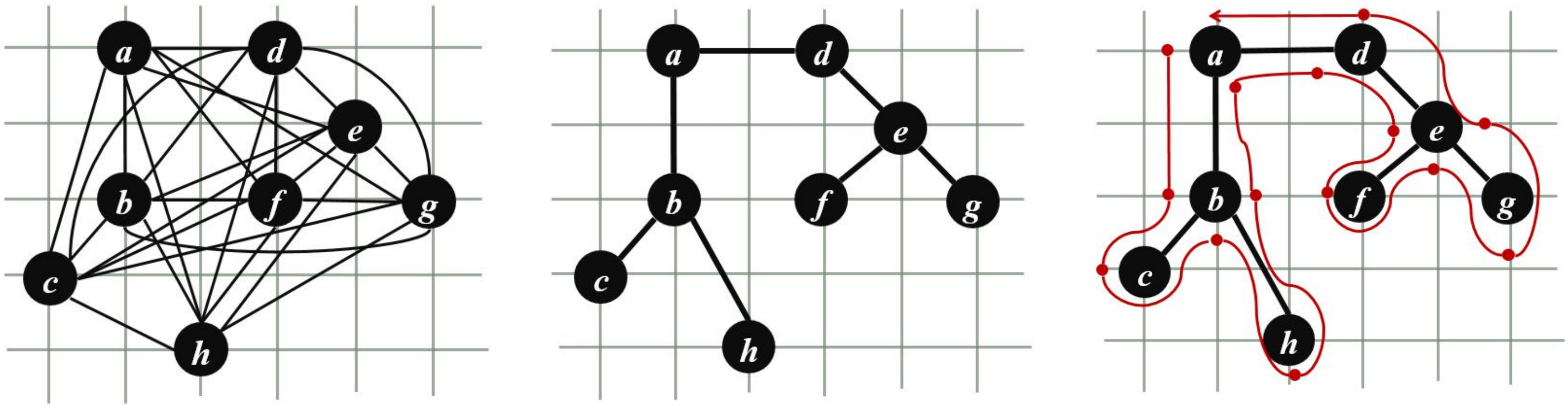
Approx-TSP-tour
Cost = 19.074



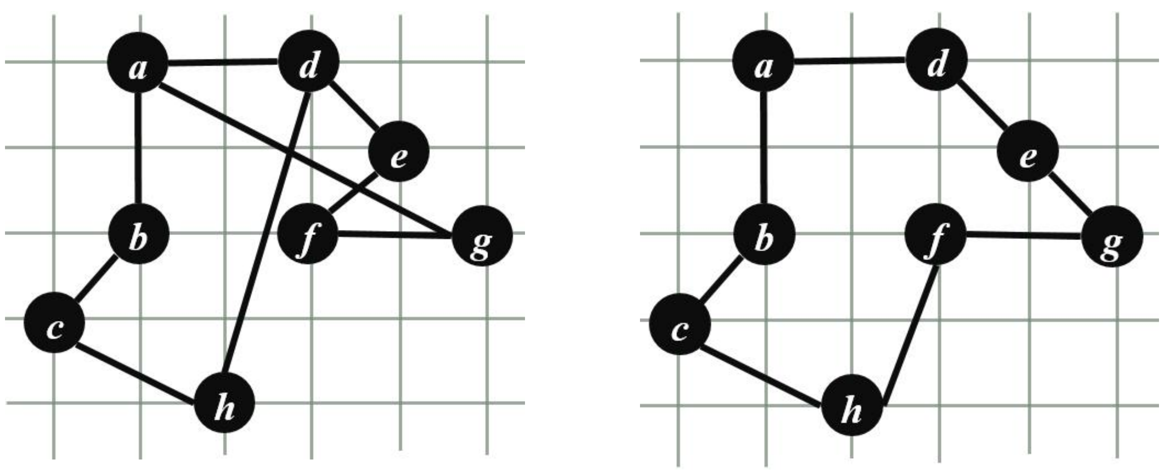
Optimal-TSP-tour
Cost = 14.715

Approximation ratio=2

APPROX-TSP-TOUR的操作过程：



(a) (b) (c)



(d) (e)

G

↓

T $\le c(H^*)$

↓

W $\le 2c(H^*)$

↓

H $\le 2c(H^*)$

Proof: *Approximation ratio*

- Let H^* denote an optimal tour for the given set of vertices and obtain a spanning tree by deleting any edge from a tour.

- The weight of the minimum spanning tree T providing ***a lower bound*** on the cost of an optimal tour:
$$c(T) \le c(H^*)$$

- Let W denote a full walk of T, since the full walk traverses every edge of T exactly twice, we have
$$c(W) = 2c(T)$$

Thus,
$$c(W) \le 2c(H^*)$$

Proof: *Approximation ratio*

- By the triangle inequality, we can delete a visit to any vertex from W and the cost does not increase.

- Let H be the cycle corresponding to preorder walk. Since H is obtained by deleting vertices from the full walk W, we have

$$c(H) \leq c(W) \leq 2c(H^*)$$

3.2 The general traveling-salesman problem

Definition

- ***The general traveling-salesman problem*** is the TSP without the assumption that the cost function c satisfies the triangle inequality.

Q: Can we find a good approximate tours in polynomial time for the general TSP ?

No, Unless P = NP!

Theorem: if $P \neq NP$, then for any constant $\rho \geq 1$, there is no polynomial-time approximation algorithm with approximation ratio ρ for the general traveling-salesman problem.

命题 q 命题 r

Proof

Theorem: if $\underline{P \neq NP}$, then for any constant $\rho \geq 1$, there is no polynomial-time approximation algorithm with approximation ratio ρ for the general traveling-salesman problem.

$$q \to r = \neg r \to \neg q$$

$\neg r$: For some number $\rho \geq 1$, there is a polynomial-time approximation algorithm A with approximation ratio ρ.

$\neg q$: P = NP

Proof：反证法

Theorem: if P $\neq$ NP, then for any constant $\rho \geq 1$, there is no polynomial-time approximation algorithm with approximation ratio ρ for the general traveling-salesman problem.

构建反例

1 Suppose to the contrary that for some number ρ ≥ 1, there is a polynomial-time approximation algorithm **A** with approximation ratio ρ.

2 $\neg q$: P = NP $\longleftrightarrow$ q: P $\neq$ NP

存在矛盾

得证！！！

① 构建特殊反例

- Let $G = (V, E)$ be an instance of the Hamiltonian cycle problem.
- Determine efficiently whether G contains a Hamiltonian cycle by making use of the hypothesized approximation algorithm A.
- Turn G into an instance of the traveling-salesman problem as follows. Let $G` = (V, E`)$ be the complete graph on V; that is,

$$E` = \{(u, v) : u, v \in V \ and \ u \neq v\}$$

① 构建特殊反例

$$E` = \{(u,v) : u,v \in V \ and \ u \neq v\}$$

$$\Downarrow$$

Assign an integer cost to each edge in E`

$$c(u,v) = \begin{cases} 1 & if\,(u,v) \in E \\ \rho|V|+1 & else \end{cases}$$

$$\Downarrow$$

Any tour that uses an edge not in E

$$(\rho|V|+1)+(|V|-1) = \rho|V|+|V| > \rho|V|$$

假设成功，可以在多项式时间内基于哈密顿回路G构建出一个求解旅行商问题的近似算法A

② 挖掘矛盾点

There is an approximation algorithm A to the TSP (已假设成功), We can find that:

- If G contains a Hamiltonian-cycle, then A must return a tour of cost no more than ρ times the cost of an optimal tour for G`.

- If G has no Hamiltonian-cycle, then A can return a tour of cost more than $\rho|V|$ for G`.

算法A能在多项式时间内找到TSP的近似解，意味着也能在多项式时间内判断出是否存在哈密顿回路

② 挖掘矛盾点

算法A能在多项式时间内找到TSP的近似解，意味着也能在多项式时间内判断出是否存在哈密顿回路（矛盾出现）

- **Theorem 34.12**: The Hamilton-cycle problem is NP-complete.

- **Theorem 34.4:** The Hamilton-cycle problem can be solved in polynomial time, then P = NP.

Conlusion: Can we find a good approximate tours in polynomial time for the general TSP ?

Unless N = NP !!!

http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/STSP.html

Optimal solutions for symmetric TSPs

When I published TSPLIB more than 10 years ago, I expected that at least solving the large problem instances to proven optimality would pose a challenge for the years to come.

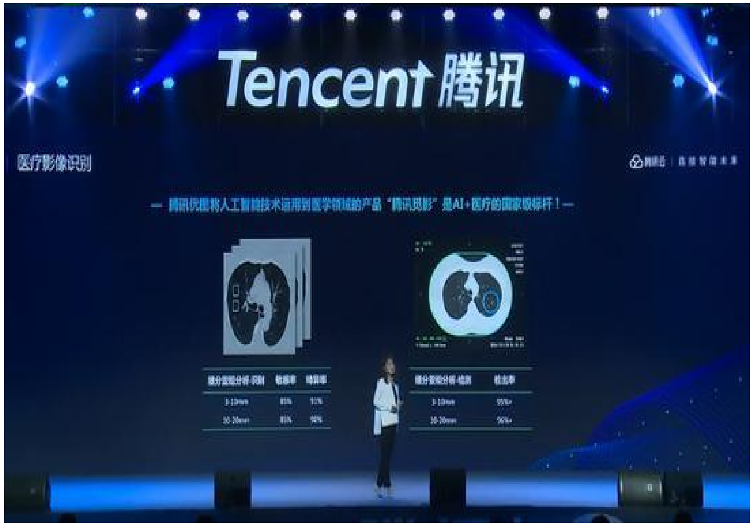
However, due to enormous algorithmic progress all problems are now solved to optimality!!

- a280 : 2579
- ali535 : 202339
- att48 : 10628
- att532 : 27686
- bayg29 : 1610
- bays29 : 2020
- berlin52 : 7542
- bier127 : 118282
- brazil58 : 25395
- brd14051 : 469385
- brg180 : 1950
- burma14 : 3323
- ch130 : 6110
- ch150 : 6528
- d198 : 15780
- d493 : 35002
- d657 : 48912
- d1291 : 50801
- d1655 : 62128
- d2103 : 80450
- d15112 : 1573084

拓展：人工智能生物医学应用

ET医疗大脑

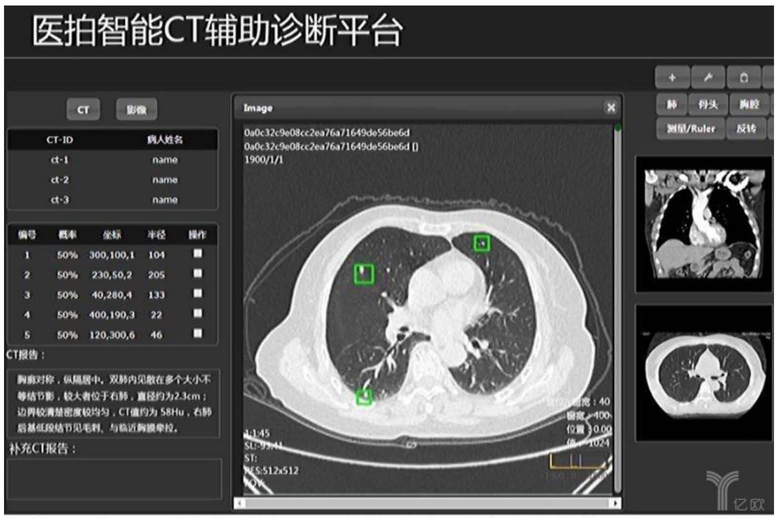
腾讯觅影



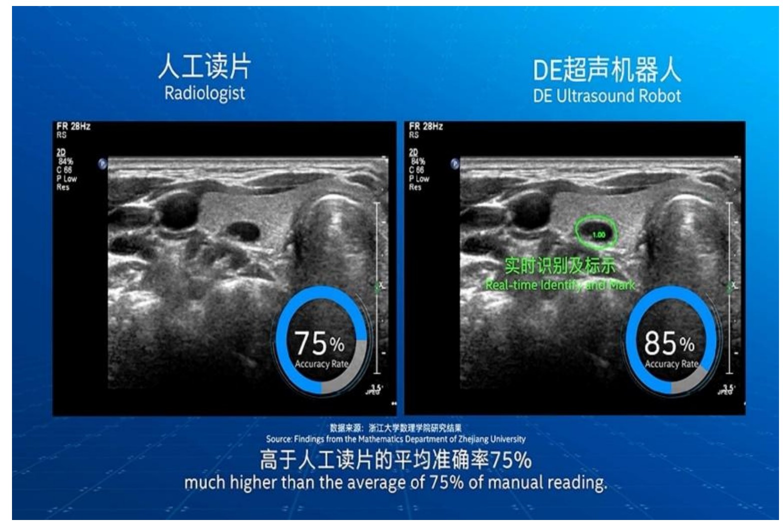
- ET医疗大脑可在患者虚拟助理 、 医学影像 、 精准医疗 、 新药研发 、 药效挖掘 、 健康管理等领域承担医生助手的角色。
- 腾讯觅影将图像识别 、 深度学习等技术与医学融合，主要开展对食管癌 、 早期肺癌 、 糖尿病性视网膜病变 、 乳腺癌等病种的早期检测 。

拓展：AI智能诊断

智能CT影像



智能超声



- 在这次新冠疫情中，多家公司研发的AI+CT影像系统，在抗疫一线都发挥了重要作用。
- 采用AI系统，在2-3秒内便实现病变区域的自动检测。

Thank You!

Q&A