Analysis of Algorithms

V. Adamchik

CSCI 570

Lecture 13

University of Southern California

Approximation Algorithms

The Design of Approximation Algorithms,
D. Williamson and D. Shmoys, Cambridge University Press, 2011.

Exam - II

Date: Thursday April 29

Time: starts at 5pm Pacific time

Length: 2 hrs and 20 mins

Time Frame: 12 hours

Locations: online, DEN Quiz

Practice: will be posted

TA Review: April 27 and April 28

Open book and notes
Use scratch paper
Follow the Honor Code (obey all rules for taking exams)
No internet searching
No talking to each other (chat, phone, messenger)

Exam - II

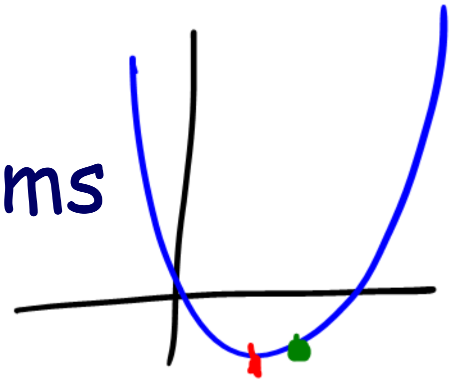
True/False

Multiple Choice

Written Problems:

- Network Flow (max flow, circulation)
- Linear Programming (standard and dual forms)
- NP Completeness (a proof by reduction)
- Approximations

Approximation Algorithms



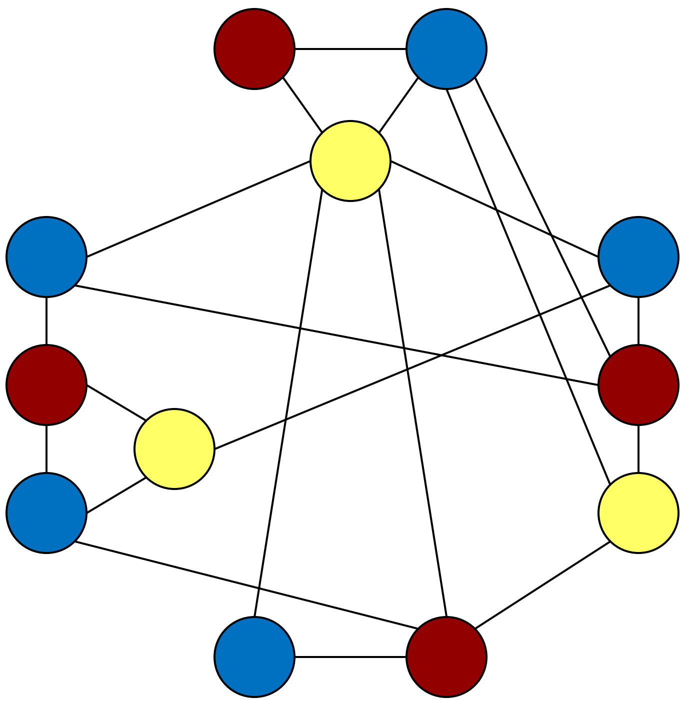
Suppose we are given an NP-hard problem to solve.

Can we develop *a polynomial-time algorithm* that produces a "good enough" solution?

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

Graph Coloring

Given a graph G=(V,E), find the minimum number of colors required to color vertices, so no two adjacent vertices have the same color.



This is NP-hard problem.

Let us develop a solution that is close enough to the optimal solution.

Greedy Approximation

Given G=(V,E) with n vertices.

Use the integers {1,2,3, …, n} to represent colors.

Order vertices by degree in descending order.

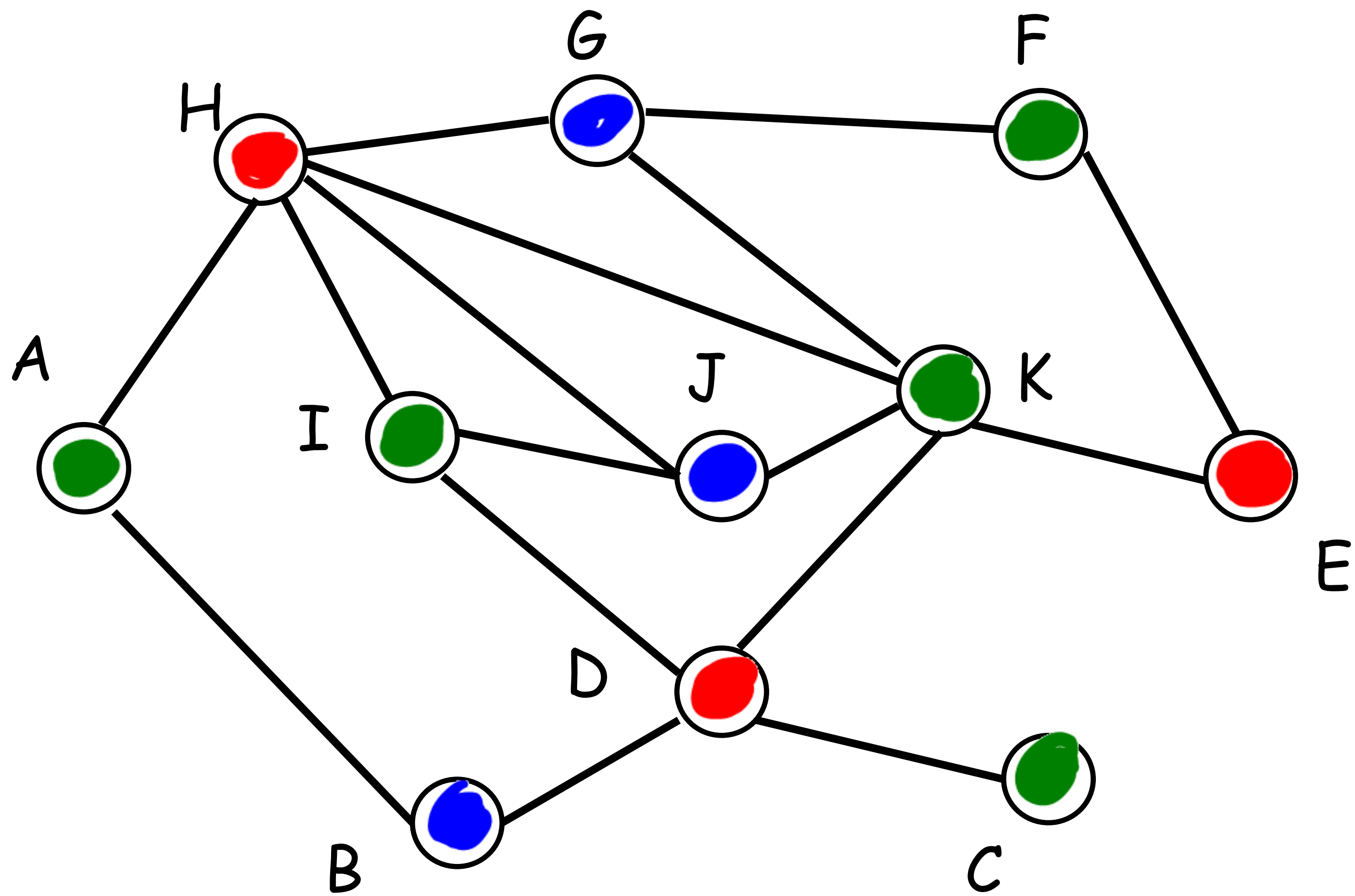
Color the first vertex (highest degree) with color 1.

Go down the vertex list and color every vertex not adjacent to it with color 1.

Remove all colored vertices from the list.

Repeat for uncolored vertices with color 2.

Example



Order: ~~H~~, ~~K~~, ~~D~~, ~~G~~, ~~I~~, ~~J~~, ~~A~~, B, ~~E~~, ~~F~~, ~~C~~

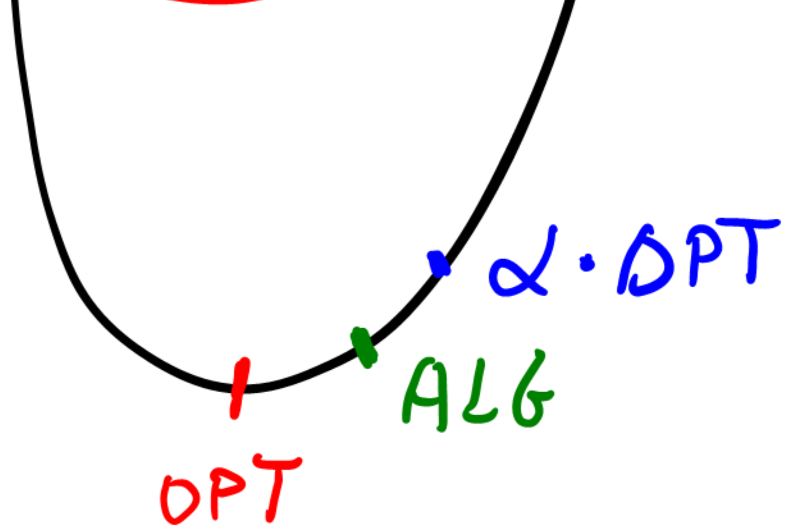
Formal Definition

convex

Let P be a minimization problem, and I be an instance of P. Let ALG(I) be a solution returned by an algorithm, and let OPT(I) be the optimal solution.

Then ALG(I) is said to be a α*-approximation* algorithm for some $\alpha > 1$, if for $ALG(I) \le \alpha \cdot OPT(I)$.

Goal: $\alpha \to 1$



$OPT \le ALG \le \alpha \cdot OPT$

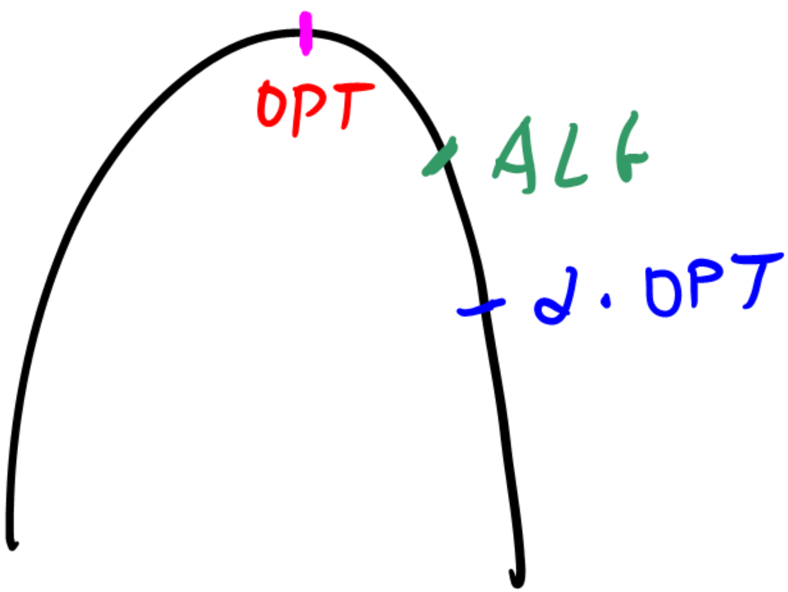
OPT = 4 bags
$\alpha = 2$

2-appr. ≤ 8 bags

Maximization Problem

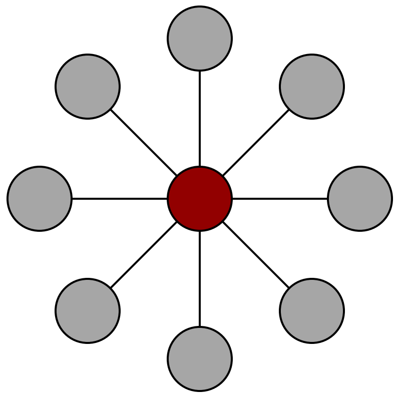
Let P be a maximization problem, and I be an instance of P. Let ALG(I) be a solution returned by an algorithm, and let OPT(I) be the optimal solution.

Then ALG(I) is said to be a α *-approximation* algorithm for some $0 < \alpha < 1$, if for ALG(I) $\geq \alpha \cdot$ OPT(I).



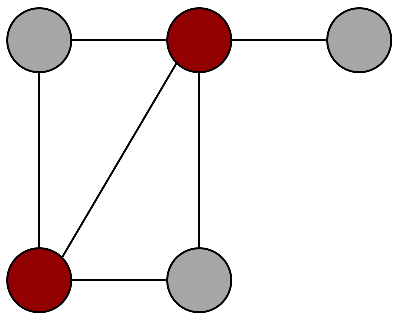
Vertex cover

Given $G=(V,E)$, find the smallest $S \subset V$ s.t. every edge is incident on a vertex in S.

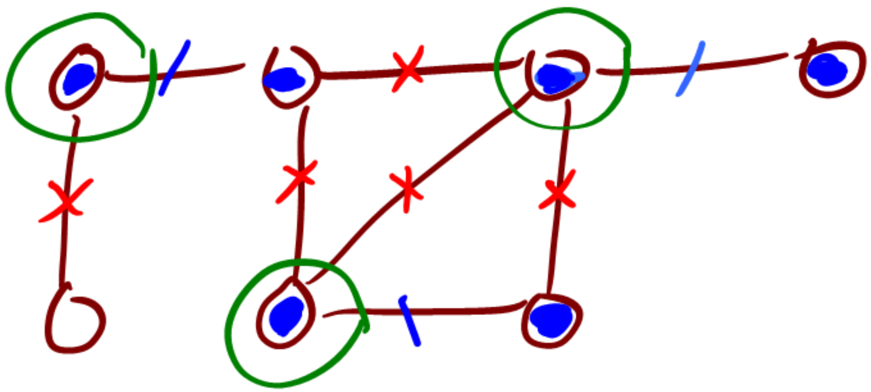


Let us design a randomized algorithm for vertex cover.

APPROX-VC(G=(V,E))
    VC ← Ø   (vertex cover)
    E' ← E(G)
    while E' ≠ Ø do
        let (u, v) be an arbitrary edge of E'
        VC ← VC U {u, v}
        remove every edge in E' incident on u or v
    return VC



Example



$VC(G) = 6$, $OPT(G) = 3$

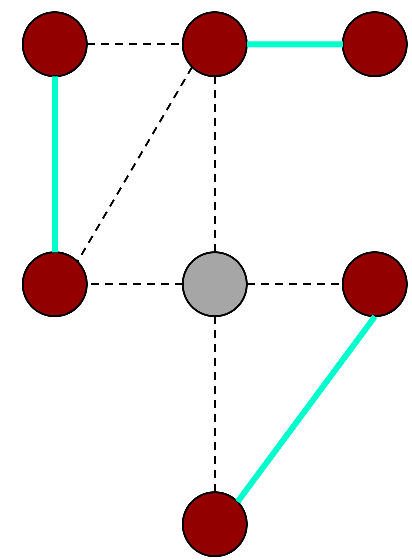
Claim. $ALG \le 2 \cdot OPT$

Prove it!

Vertex Cover

Lemma. Let M be a matching in G, and VC be a vertex cover, then

$$|VC| \geq |M|$$



Proof.

VC must cover each edge in M

2-Approximation Vertex Cover

not maximum

Q: $\alpha < 2$?

Approx-VC(G):
M - maximal matching on G
VC - take both endpoints of edges in M
Return VC

Theorem. Let OPT(G) be the size of the optimal vertex cover and VC = ApproxVC(G). Then $|VC| \le 2 \cdot OPT(G)$.

our algorithm

Proof. $|VC| = 2 \cdot |M| \le 2 \cdot OPT$

↑ Lemma

Can we do better than 2-Approximation?

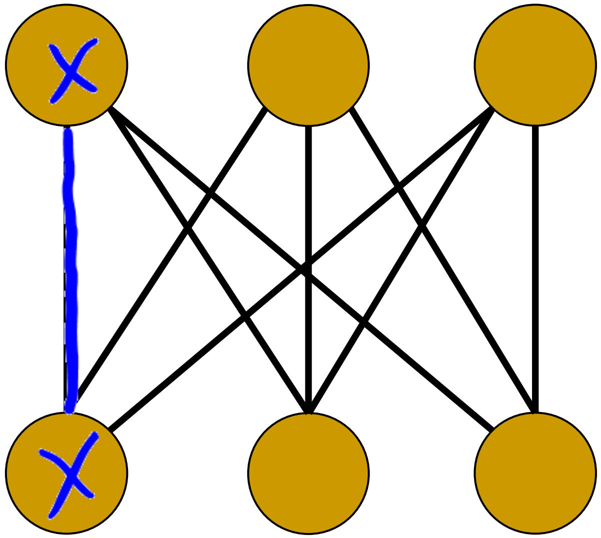
is tight

Consider $K_{n,n}$

$|OPT\ VC(K_{n,n})| = n$

$|maximum\ matching| = n$

$K_{3,3}$



$ALF\ VC(K_{n,n}) = 2 \cdot n$

Traveling ~~Salesman~~ Problem

person

The traveling salesman problem consists of a salesman and a set of cities (a complete weighted graph). The salesman has to visit each one of the cities starting from a certain one (e.g. the hometown) and returning to the same city. The challenge of the problem is that the traveling salesman wants to minimize the total length of the trip.

Theorem.
The traveling salesman problem is NP-complete.

NP-hard.

Proof

Given K.

First, we have to prove that TSP belongs to NP.

Secondly, we prove that TSP is NP-hard.

$HC \leq_p TSP$

$G' = (V, E')$

construction

$\forall G,\ HC(G) \longrightarrow G',$ 1) make complete 2) weights $\notin G$

$G = (V, E)$

$$w(u,v) = \begin{cases} 0, & (u,v) \in E \\ 1, & (u,v) \notin E \end{cases}$$

Claim. G has a HC iff, G' has a TSP tour of cost zero.

$\Rightarrow$ construction.

$\Leftarrow$

Solving TSP

Given an undirected complete graph $G=(V,E)$ with edge cost $c:E \in R^+$, find a min cost Hamiltonian cycle.

We will consider a metric TSP. L_2

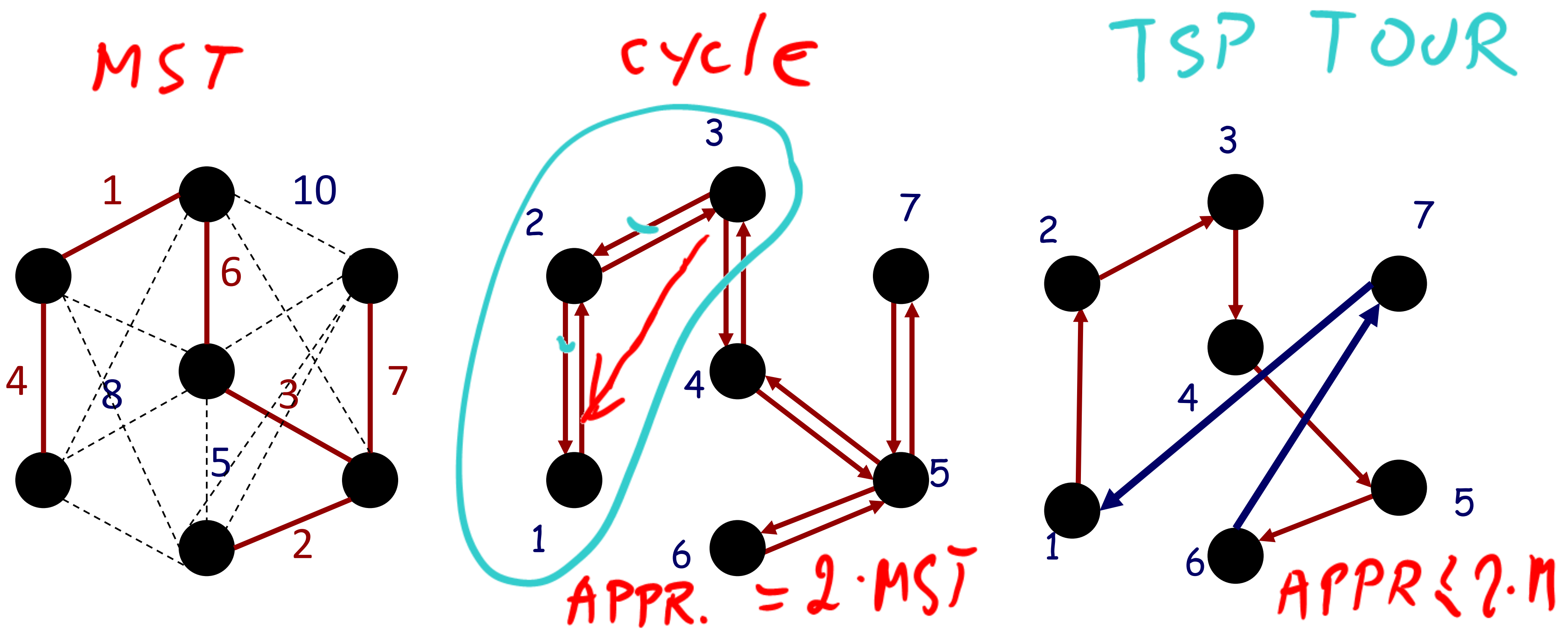
In the metric TSP, edge costs have to satisfy the triangle inequality, i.e. for any three vertices x, y, z, $c(x,z) \leq c(x,y) + c(y,z)$.

We develop approximation greedy algorithm.

Approximation Greedy Algorithm

Approx-TSP(G):

1) Find a MST of G (complete graph)
2) Create a cycle by $\boxed{\text{doubling}}$ edges
3) Remove double visited edges



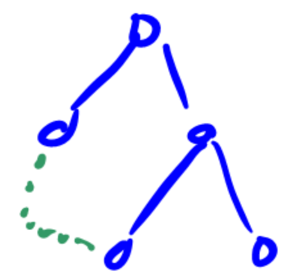
Approximation TSP

Q: $\alpha < 2$

Theorem. Approx-TSP is a 2-approximation algorithm for a metric TSP.

Proof.

$$|\text{Approx-TSP}| \le 2\cdot\overset{\text{tree}}{|\text{MST}|} \le 2\cdot\overset{\text{cycle}}{|\text{OPT}|}$$



triangle inequality

we can get a spanning tree from HC by removing edges

Christofides Algorithm

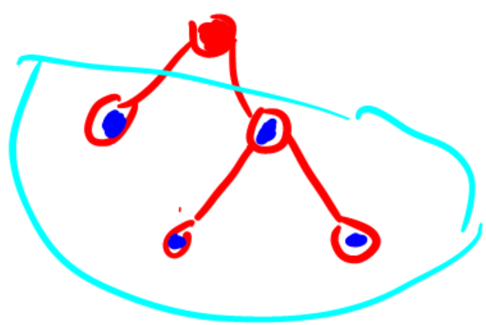
$\in$ CMU

Observe that a factor 2 in the approximation ratio is due to doubling edges.

But any graph with even degrees vertices has an Eulerian cycle. (A Eulerian path visits each edge exactly once)

Thus we have to add edges only between odd degree vertices

min-cost matching



Christofides Algorithm

Theorem (without proof).

Christofides is 3/2 approximation for Metric TSP

The algorithm has been known for over 40 years and yet no improvements have been made since its discovery.

General TSP

Theorem. If P$\neq$NP, then for $\forall\alpha>1$ there is NO a polynomial time α-approximation of general TSP.

Proof.

Suppose for contradiction that such an α -approximation algorithm exists. We will use this algorithm to solve the Hamiltonian Cycle problem.

Start with $G=(V,E)$ and create a new complete graph G' with the cost function

$$c(u,v) = 1, \text{ if } (u,v)\in E$$
$$c(u,v) = \alpha\cdot V+1, \text{ otherwise}$$

General TSP

Proof (continue)

$c(u,v) = 1$, if $(u,v) \in E$
$c(u,v) = \alpha \cdot V+1$, otherwise

1) If G has HC, then $|TSP(G')| = \boxed{V.}$

2) If G has no HC, then

$$|TSP(G')| \geq \underbrace{(V - 1)}_{\in E} + \underbrace{\alpha \cdot V + 1}_{\notin E} = V + \alpha \cdot V \boxed{\geq \alpha \cdot V}$$

Since the |TSP| differs by a factor α, our approx. algorithm can be able to distinguish between two cases, thus decides if G has a ham-cycle. Contradiction.

Break, 10 mins | to $P \neq NP$

Bin Packing

You have an infinite supply of bins, each of which can hold M maximum weight. You also have n objects, each of which has a weight w_i (w_i is at most M). Our goal is to partition the objects into bins, such that we use as few bins as possible. This problem in general is NP-hard.

Devise a 2-approximation algorithm.

online algo

First-fit approach

1. process items in the original order
2. if an item doesn't fit the first bin, put it into the next bin
3. if an item does not fit into any bins, open a new bin

Runtime-? $O(n^2)$

Example,

items = {~~40~~, ~~20~~, ~~35~~, ~~15~~, ~~25~~, ~~5~~, ~~30~~, ~~10~~}

M = 45

ALG = 5
OPT = 4

Bin 1	Bin 2	Bin 3	Bin 4	Bin 5
40, 5	20, 15 10	35	25	30

last

$ALG = m$ $\quad m \le 2 \cdot m^*$, $ALG \le 2 \cdot OPT$

$OPT = m^*$

Proof: a 2-approximation

Suppose we use m bins.

Let m* denote the optimal number of bins.

Clearly, $m^* \geq (\sum w_i)/M$, ? we cannot break items

On the other hand, $(\sum w_i)/M > (m-1)/2$

This follows form the fact that at least (m-1) bins are more than half full.

impossible

Because, if there are two bins less than half full, items in the second bin will be placed into the first by our algorithm.

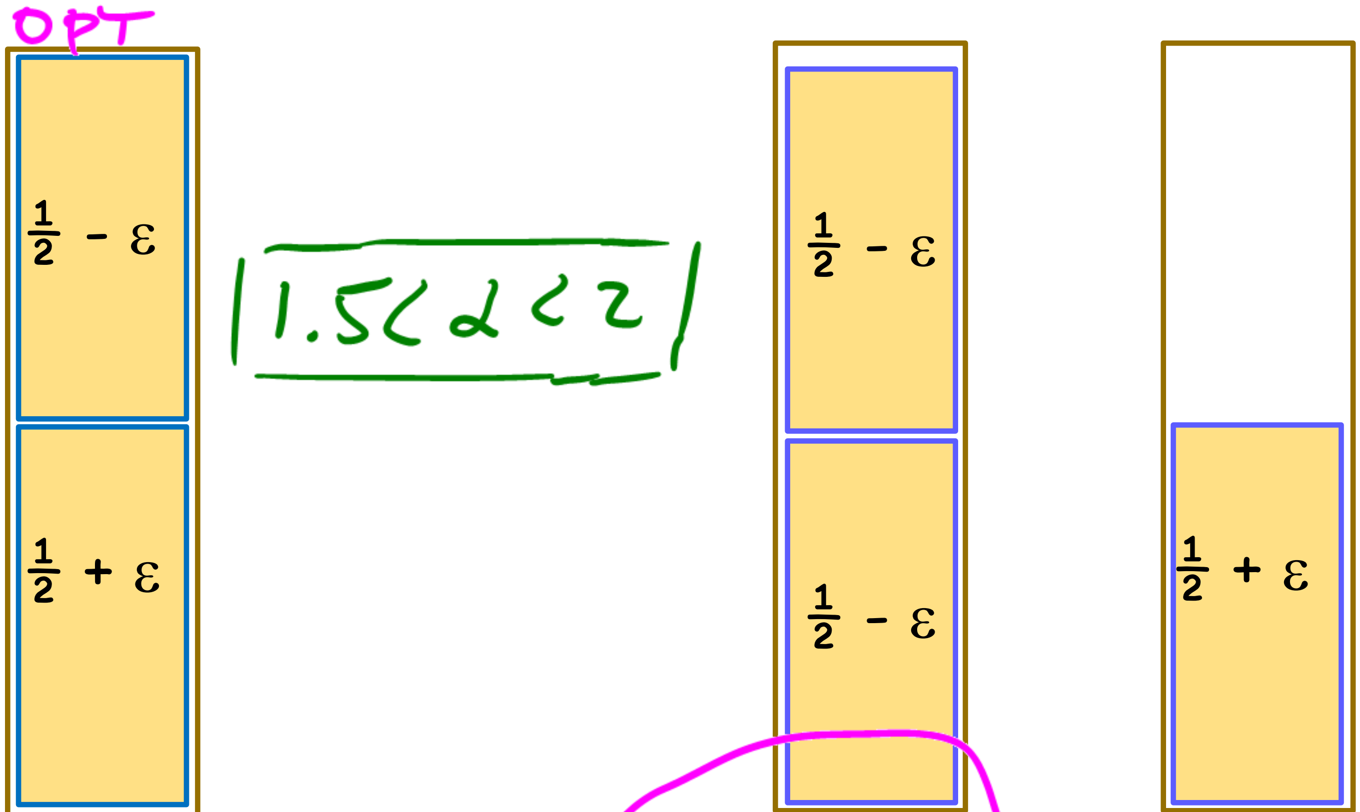
We have showed, $m^* \geq (\sum w_i)/ M > (m-1)/2$

Comparing the left and right hand sides, we derive

$$2m^* > m-1 \qquad \text{or,} \qquad \boxed{m \leq 2m^*}$$

$1 < ? < \alpha < 2$

Developing a lower bound (M=1)



$\boxed{1.5 < \alpha < 2}$

Insert 2N items: $\frac{1}{2} - \varepsilon, \ldots, \frac{1}{2} - \varepsilon, \frac{1}{2} + \varepsilon, \ldots, \frac{1}{2} + \varepsilon$ $\quad \varepsilon \to 0$

OPT: N bins

FF: N/2 bins + N bins = 1.5 N

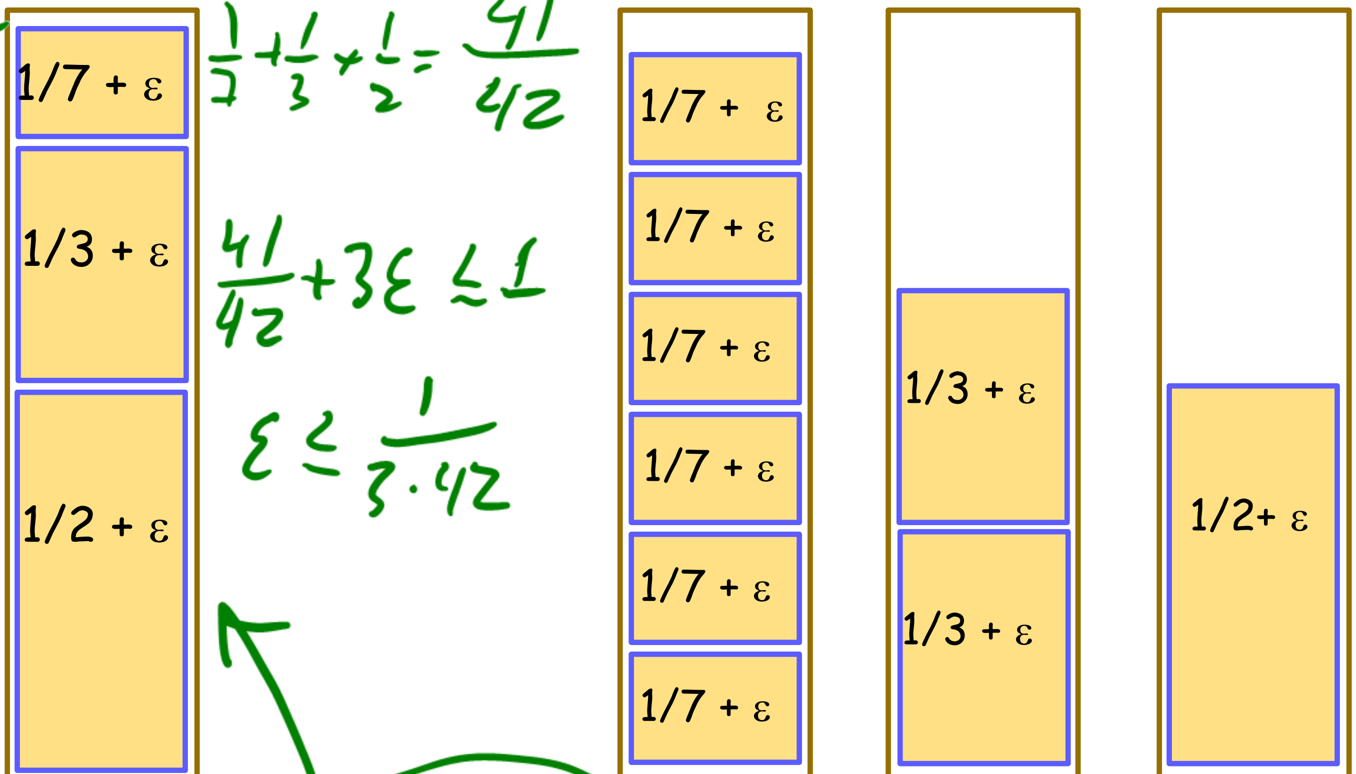
$\alpha = 1.5$

FF ≥ 1.5 OPT

$\frac{1}{42}$

Developing a lower bound (M = 1)

gap



$\frac{1}{7}+\frac{1}{3}+\frac{1}{2}=\frac{41}{42}$

$\frac{41}{42}+3\varepsilon \leq 1$

$\varepsilon \leq \frac{1}{3\cdot 42}$

$\varepsilon \to 0$

$\varepsilon > 0$

3N items: 1/7 + ε,…, 1/7 + ε, 1/3 + ε,…, 1/3 + ε, ½ + ε,…, ½ + ε

OPT: N bins

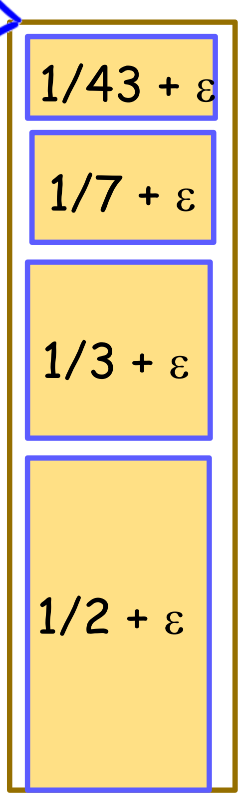
FF: N/6 + N/2 + N bins = 5/3 N

$1.66 < \alpha < 2$

FF ≥ 1.66 OPT

Developing a lower bound

1/1806
gap



Observe

1 - (1/2+1/3+1/7) = 1/42

$$1-\left(\frac{1}{2}+\frac{1}{3}+\frac{1}{7}+\frac{1}{43}\right)=\frac{1}{1806}$$

4N items: 1/43 + ε,..., 1/43 + ε,
1/7 + ε, ... , 1/7 + ε,
1/3 + ε, ... , 1/3 + ε,
$\frac{1}{2}$ + ε,. . ., $\frac{1}{2}$ + ε

OPT: N bins

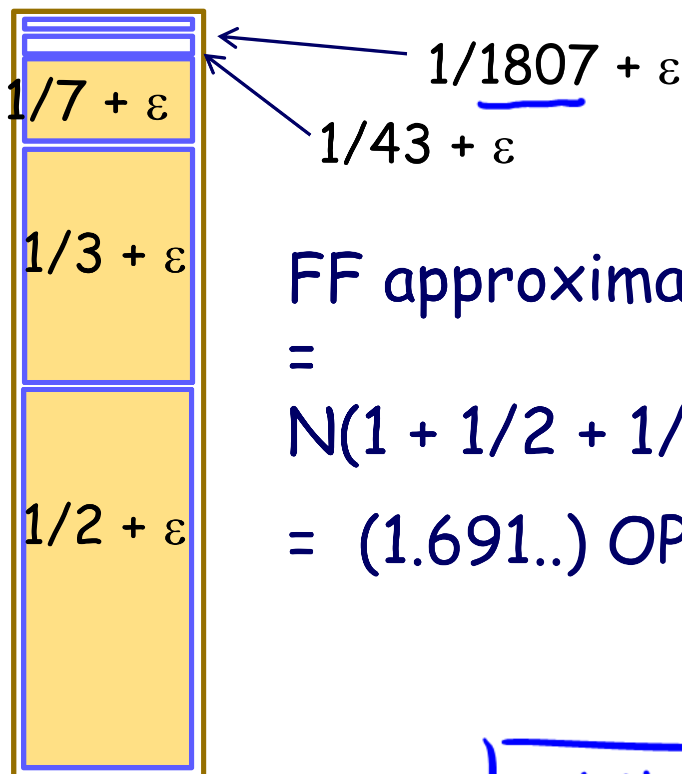
FF: N/42 + N/6 + N/2 + N = 71/42 N

FF ≥ 1.6905 OPT

$$\boxed{1.69 < \alpha < 2}$$

Developing a lower bound

Observe, 1 - (1/2+1/3+1/7+1/43) = 1/1806



OPT: N bins

FF approximation

=

N(1 + 1/2 + 1/6 + 1/42 + 1/1806 + …)

= (1.691..) OPT

Theorem (1976)

FF *This is a 17/10-approximation algorithm.*

In the limit, the ratio is 1.7

$\boxed{1.691 < \alpha < 2}$

$\alpha \approx 1.7$

offline algo

Sorted First-fit approach

1. sort the input in descending order
2. apply the first-fit algorithm

> **Theorem.**
> *This is a 11/9 - approximation algorithm.*

Alg = 4
OPT = 4

Example,

items = {40, 20, 35, 15, 25, 5, 30, 10}

M = 45   40, 35, 30, 25, 20, 15, 10, 5

Bin 1	Bin 2	Bin 3	Bin 4	Bin 5
40	35	30	25	
5	10	15	20	

Discussion Problem

Suppose you are given a set *A* of positive integers $a_1, a_2, \ldots, a_n$ and a positive integer *B*. A subset $S \subseteq A$ is called *feasible* if the sum of the numbers in *S* does not exceed *B*. You would like to select a feasible subset $S \subseteq A$ whose total sum is as large as possible. Give a linear-time algorithm that finds a feasible set $S \subseteq A$ whose total sum is at least half as large as the maximum total sum of any feasible set. You may assume that $a_i \le B$.

Example: If $A = \{8, 2, 4, 3, 5\}$ and $B = 11$ then the optimal solution is the subset $S = \{8, 3\}$. $8+3=11$

$ALG \geq B/2$

Algorithm:

1 Add items from A (in given order) to some set X, until the total sum of $X \le B$

$X = \{8, 2\}$

Kevin

$A = \{4, 2, 8, 3, 5\}$

$X = \{4, 2\}$

$Y = \{8\}$

2. Let the next item belong to another set $\underline{Y}$.

$Y = \{4\}$

solution: either X or $\underline{Y}$.

Note, $X + Y > B$

one of them $\geq B/2$