Algorithmics
Tutorial 11

Jo Grundy

ECS Southampton

May 13, 2022

Tutorial

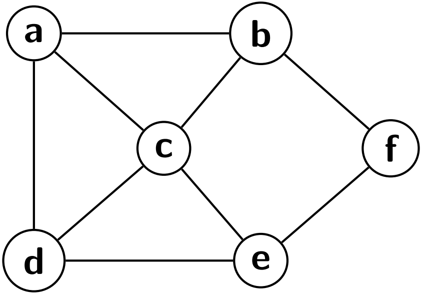
Problems:

- Backtracking solution for Hamiltonian Cycle
- knapsack problem solution with branch and bound
- knapsack using genetic algorithm

Backtracking search

Tutorial - Backtracking

Q: Given an undirected graph such as the following:



describe a *backtracking strategy* for finding a Hamiltonian cycle (a tour which visits each vertex exactly once) in the graph.

Tutorial - Backtracking

Representation?

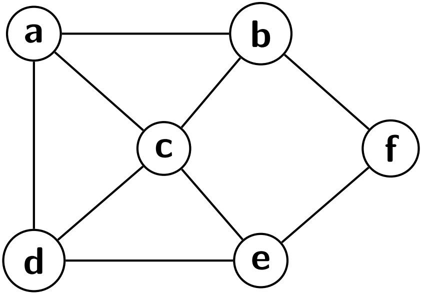
Tutorial - Backtracking

Representation?

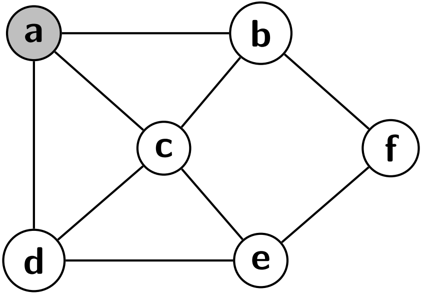
- vector giving order of notes visited
- $[S, 0, 0, 0, 0, 0, S]$
- S is start and end
- 0 is undecided

For example: $[a, b, c, d, e, f, a]$

Tutorial - Backtracking

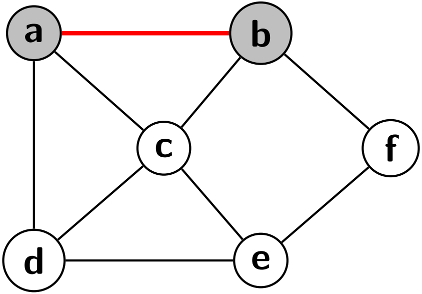


Tutorial - Backtracking



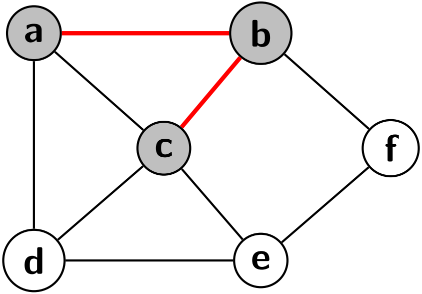
$$(\boldsymbol{a}, 0, 0, 0, 0, 0, \boldsymbol{a})$$

Tutorial - Backtracking



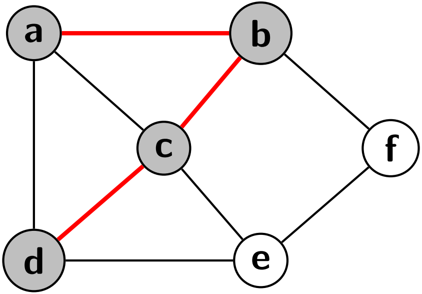
$(\boldsymbol{a}, \boldsymbol{b}, 0, 0, 0, 0, \boldsymbol{a})$

Tutorial - Backtracking



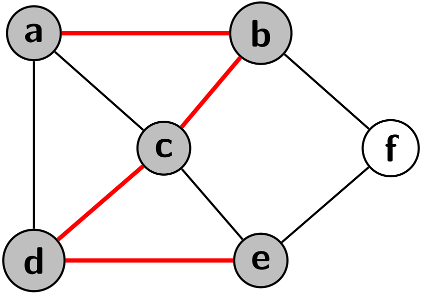
$$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, 0, 0, 0, \boldsymbol{a})$$

Tutorial - Backtracking



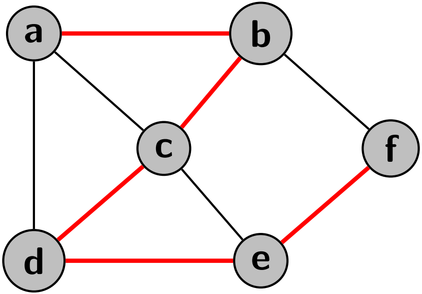
$$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, 0, 0, \boldsymbol{a})$$

Tutorial - Backtracking



$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}, 0, \boldsymbol{a})$

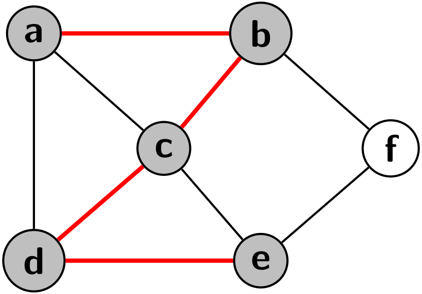
Tutorial - Backtracking



$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}, \boldsymbol{f}, \boldsymbol{a})$

X - Backtrack!

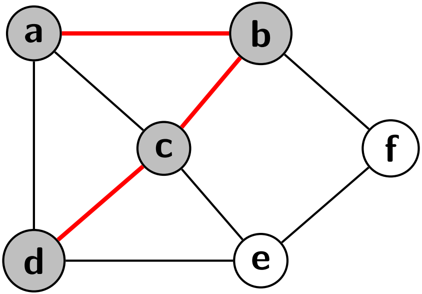
Tutorial - Backtracking



$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}, 0, \boldsymbol{a})$

Backtracking..

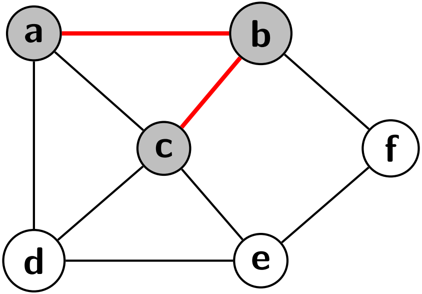
Tutorial - Backtracking



$$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, 0, 0, \boldsymbol{a})$$

Backtracking..

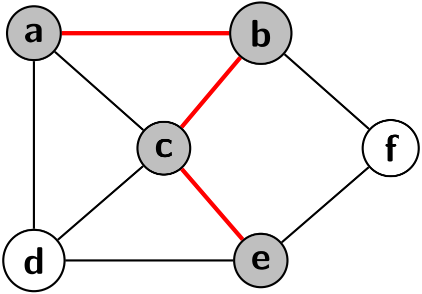
Tutorial - Backtracking



$$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, 0, 0, 0, \boldsymbol{a})$$

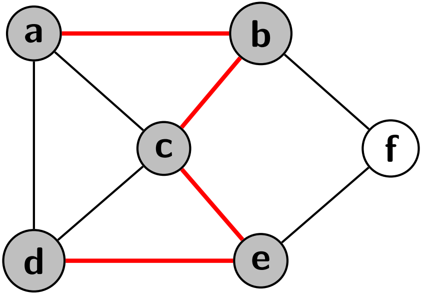
Backtracking..

Tutorial - Backtracking



$$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{e}, 0, 0, \boldsymbol{a})$$

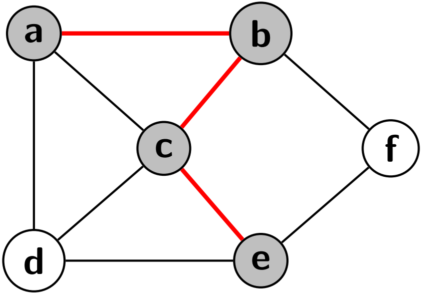
Tutorial - Backtracking



$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{e}, \boldsymbol{d}, 0, \boldsymbol{a})$

X - Backtrack!

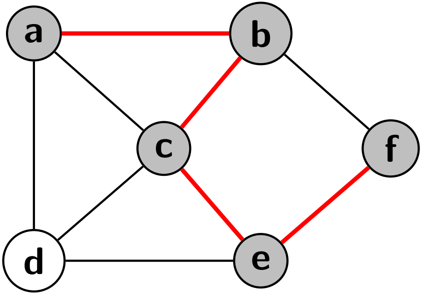
Tutorial - Backtracking



$$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{e}, 0, 0, \boldsymbol{a})$$

Backtracking..

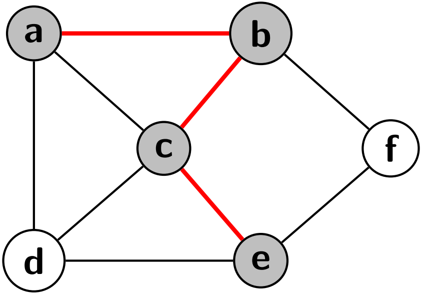
Tutorial - Backtracking



$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{e}, \boldsymbol{f}, 0, \boldsymbol{a})$

X - Backtrack!

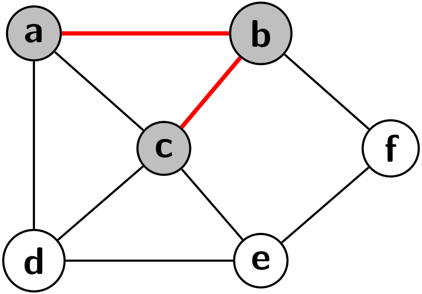
Tutorial - Backtracking



$$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{e}, 0, 0, \boldsymbol{a})$$

Backtracking..

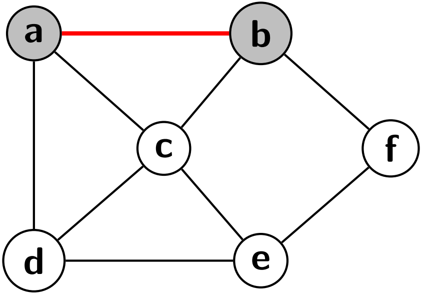
Tutorial - Backtracking



$$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, 0, 0, 0, \boldsymbol{a})$$

Backtracking..

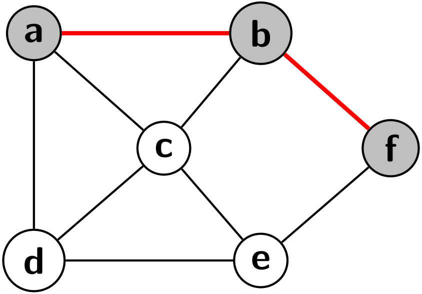
Tutorial - Backtracking



$$(\boldsymbol{a}, \boldsymbol{b}, 0, 0, 0, 0, \boldsymbol{a})$$

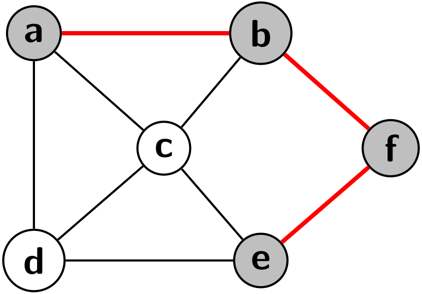
Backtracking..

Tutorial - Backtracking



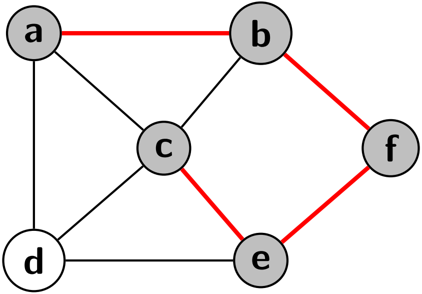
$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{f}, 0, 0, 0, \boldsymbol{a})$

Tutorial - Backtracking



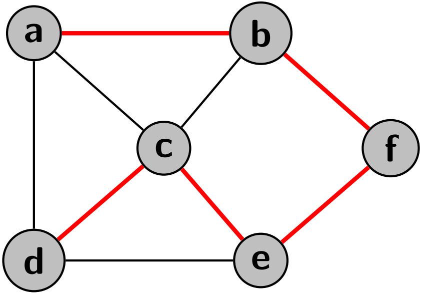
$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{f}, \boldsymbol{e}, 0, 0, \boldsymbol{a})$

Tutorial - Backtracking



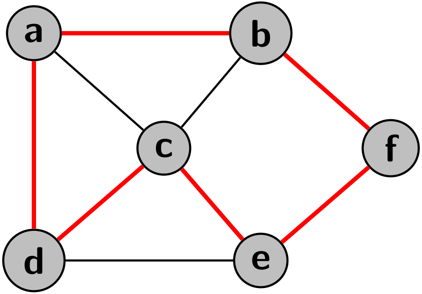
$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{f}, \boldsymbol{e}, \boldsymbol{c}, 0, \boldsymbol{a})$

Tutorial - Backtracking



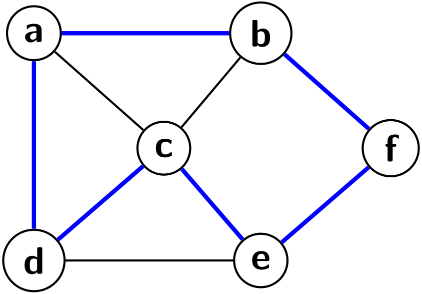
$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{f}, \boldsymbol{e}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{a})$

Tutorial - Backtracking



$$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{f}, \boldsymbol{e}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{a})$$

Tutorial - Backtracking



$$(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{f}, \boldsymbol{e}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{a})$$

Tutorial - Backtracking

Backtracking strategy involves:

- deciding a representation
- a function that encodes the constraints in the problem
- it must be able to evaluate partial solutions
- a way to enumerate the partial solutions

See code
`https://github.com/jogrundy/algo_notebooks/tree/main`

Tutorial - Branch and Bound

Cut the search tree off if answer cannot be better than one already found

Knapsack problem

- Optimisation
- find the **best** valid solution

item	weight	value
x_1	24	100
x_2	14	4
x_3	39	30
x_4	60	60
x_5	98	200
x_6	22	95
x_7	5	5
x_8	32	10
x_9	16	20
x_{10}	9	12

Tutorial - Branch and Bound

Representation?

Tutorial - Branch and Bound

Representation?

- $$[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$
- binary vector, 1 if included, 0 if not

If using brute force, how many possible solutions?

Tutorial - Branch and Bound

Representation?

- $$[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$
- binary vector, 1 if included, 0 if not

If using brute force, how many possible solutions? $2^1 0$
Strategy?

- backtracking dfs
- branch and bound

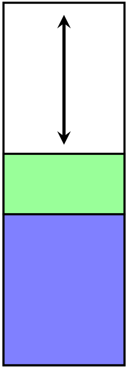
Tutorial - Branch and Bound

Branch and bound needs an **upper bound** to be calculable on partial solutions

How?

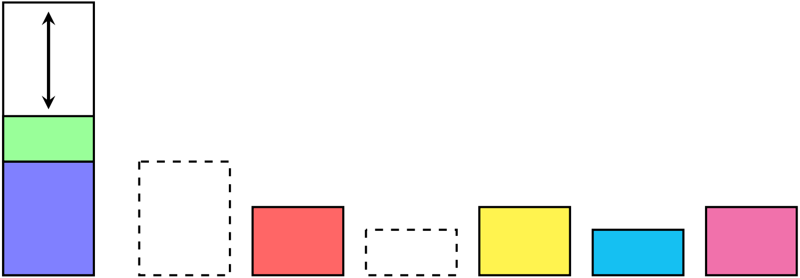
Tutorial - Branch and Bound

Branch and bound needs an **upper bound** to be calculable on partial solutions
How?
Could consider residual capacity?



Tutorial - Branch and Bound

Could use highest ratio of value to weight of the objects remaining with the residual capacity to calculate what the best possible value it might have could be..



See code
`https://github.com/jogrundy/algo_notebooks/tree/main`

Tutorial - Genetic Algorithms

Can you solve knapsack problem with a genetic algorithm?

Tutorial - Genetic Algorithms

Can you solve knapsack problem with a genetic algorithm?
Yes! - though optimality not guaranteed

- representation? - 00101101
- fitness function? - lots to chose from
- $$f(x) = \begin{cases} 0 & \text{if } w \cdot x \leq 150, \\ -\infty & \text{otherwise.} \end{cases}$$
- $$f(x) = \begin{cases} totalValue & \text{if } w \cdot x \leq 150, \\ 0 & \text{otherwise.} \end{cases}$$

Tutorial - Genetic Algorithms

- Selection? Roulette Selection - ?

Tutorial - Genetic Algorithms

- Selection? Roulette Selection - ?
- Calculate total fitness for whole population
- Divide fitness for each individual by total
- Mutation? - bit flip?
- Crossover? - Uniform?