# Gale-Shapley – Stable marriage (Labb 1)

Initialize each person to be free.

while (some man is free and hasn't proposed to every woman) {

Choose such a man m

w = 1st woman on m's list to whom m has not yet proposed

f (w is free)

assign m and w to be engaged

else if (w prefers m to her fiancé m')

assign m and w to be engaged, and m' to be free

else

w rejects m

}

**0(n^2)**

# Breadth-first search (BFS) (Graphs) – Word ladders (Labb 2)

* Undersöker utåt från startnoden alla möjliga vägar, lägger till noder ett lager åt gången.

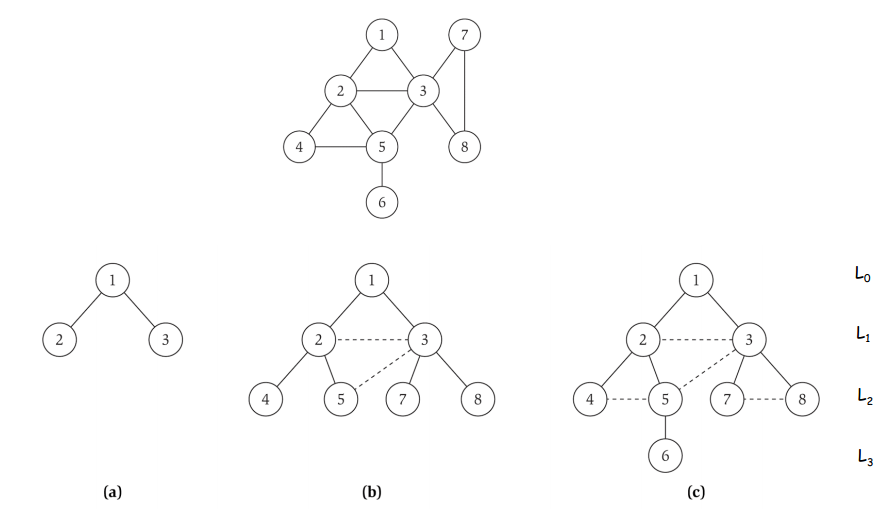
**BFS algorithm**

L0 = { s }.

L1 = all neighbors of L0

L2 = all nodes that do not belong to L0 or L1, and that have an edge to a node in L1

Li+1 = all nodes that do not belong to an earlier layer, and that have an edge to a node in Li



**O(n+m)**  (n-nodes, m-arcs) eller **O(n^2)**

#### Graph Traversal

*A graph G is a simple way of encoding pairwise relationships among a set of objects. It consists of a collection V of nodes and a collection E of edges, each of which joins two of the nodes. We thus represent an edge e E as a two-element subset of V : e = {u,v} for some u,v  in V where we call u,v the ends of e.*

*Trees can’t cycle by the way.*

Examples: BFS, DFS, finding cycles within a graph, maze problems

# Greedy algorithms – Spanning USA (Labb 3)

* Betrakta jobben i någon viss ordning/sortering.
* Ta ut varje jobb för att jämföra det med de som redan är tagna. Beroende på vilken resultat som man vill åt sorteras jobben på olika sätt.

An algorithm is greedy if it builds up a solution in small steps, choosing a decision at each step greedily to optimize some underlying criterion.

Klassiska exempel är ”shortest path in graph”, ”minimum spanning tree problem” och ” Interval Scheduling”.

**Prim’s** or **Kruskal’s** algorithm (eller **Dijkstras algoritm**).

**Kruskal’s algorithm**

* För att hitta ett minimalt uppspännande träd T i den sammanhängande grafen G
* Upprepa tills T innehåller alla noder i G
  + Låt v vara den kortaste kanten i G som inte märkts som förbrukad
  + Märk v som förbrukad
  + Om v inte bildar en cykel i T
    - Lägg v till T
* T är ett minimalt uppspännande träd

**O(mlogn)** (n-nodes, m-arcs)

# Divide and conquer – Closest pair in a plane (Labb 4)

* Dela upp problemet i mindre delar
* Lös varje del rekursivt.
* Kombinera lösningarna till subproblem till en slutgiltig lösning.

Klassiska exempel är ”Quicksort”, ”Mergesort” och ”Closest pair of points”.

Algorithm – Closest pair

* Divide: draw vertical line L so that roughly ½n points on each side.
* Conquer: find closest pair in each side recursively.
* Combine: find closest pair with one point in each side.
* Return best of 3 solutions.

**O(nlogn)**

Mergesort – O(nlogn)

# Dynamic programming - [Me: Gorilla or sea cucumber?](http://fileadmin.cs.lth.se/cs/Education/EDAF05/Labbar/cucumber.pdf) (Labb 5)

Almost a brute force solution (n2) where one saves away all previous calculations in a matrix.

Bra på att lösa optimeringsproblem.

Gissa + rekursion + minne (spara gamla beräkningar)

DP typ ”försiktig” bruteforce

Exempel: ”Weighted Interval Scheduling” från föreläsningarna, tv-program.

**Sequence allignment**

Def. OPT(i, j) = min cost of aligning strings x1 x2 . . . xi and y1y2 . . . yj

Case 1: OPT matches xi-yj.

– pay mismatch for xi-yj+ min cost of aligning two strings x1x2. . . xi-1 and y1y2 . .yj-1

Case 2a: OPT leaves xi unmatched.

– pay gap for xi and min cost of aligning x1x2. . . xi-1 and y1y2. . . yj

Case 2b: OPT leaves yj unmatched.

– pay gap for yj and min cost of aligning x1x2 . . . xi and y1y2 . . . yj-1

**O(nm)** (n/m längden av strängarna som jämförs) eller O(n^2)

### Network flow – Railroad planning (Labb 6)

Similar to graph traversal. Is used for matching of two objects or more to find an optimal solution by pushing “flow” through the network/graph.

Examples: Table arrangement, matching in a bipartite graph, min cut maximal flow, image segmentation

**Flow value lemma**. Let f be any flow, and let (A, B) be any s-t cut.

Then, the net flow sent across the cut is equal to the amount leaving s.

**O(m^2logC)**

### NP-completeness

Ofta problem med flera olika delproblem. Tex att man ska hitta en viss väg men får inte besöka vissa platser/noder.

**Recipe to establish NP-completeness of problem Y.**

* Step 1. Show that Y is in NP.
* Step 2. Choose an NP-complete problem X.
* Step 3. Prove that X ≤ p Y.

#### Partitioning problems

**Graph colouring**

Om grafen kan färgas med k olika färger och ”independent set” gäller.

Is it possible to color each node in a graph G with the condition that neighbouring nodes in G are not allowed to have the same color?

**3-dimensional matching**

Som bipartite matchning men 3-dimensionellt.

Given three sets of X, Y and Z, each of size n, and given a set TX YZ of allowed combinations of triples, does there exist a set of n triples in T so that each element of XYZ is contained in exactly one of these triples.

#### Covering problems

**Vertex cover**

Given a graph G = (V, E) and an integer k, is there a subset of vertices S ⊆ V such that |S| ≤ k, and for each edge (u, v) either u ∈ S, or v ∈ S, or both.

**Set cover**

Given a set U of n elements and a collection S1,....Sm of subsets of U, and a number k, does there exist a collection of at most k of these sets whose union is equal to all of U?

#### Packing problems

**Independent set**

Given a tree, find a maximum cardinality subset of nodes such that no two share an edge. (Hur många noder som kan färgas utan att två färgade noder är sammankopplade (en mängd)).

**Set packing**

Suppose we have a finite set *S* and a list of subsets of *S*. Then, the set packing problem asks if some *k* subsets in the list are pairwise disjoint (in other words, no two of them intersect)

(Given a set U of n elements and a collection S1,....Sm of subsets of U, and a number k, does there exist a collection of at least k of these sets with the property that no two of them intersect?)

#### Sequencing problems

**Hamiltonian Cycle:**

Find a cycle through a graph G by visiting all nodes in G only once.

**Hamiltonian Path:**

Find a simple path through a graph G visiting all nodes.

**Travelling salesman problem**

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?

(Visit a maximal number of nodes within a weighted graph with a fixed bound of allowed “travel” weight.)

#### Numerical Problems

**Subset sum:**

Given natural numbers w1,.....,wn, and a target number W, is there a subset of {w1,....,wn} that adds up to precisely W?

**Knapsack problem**

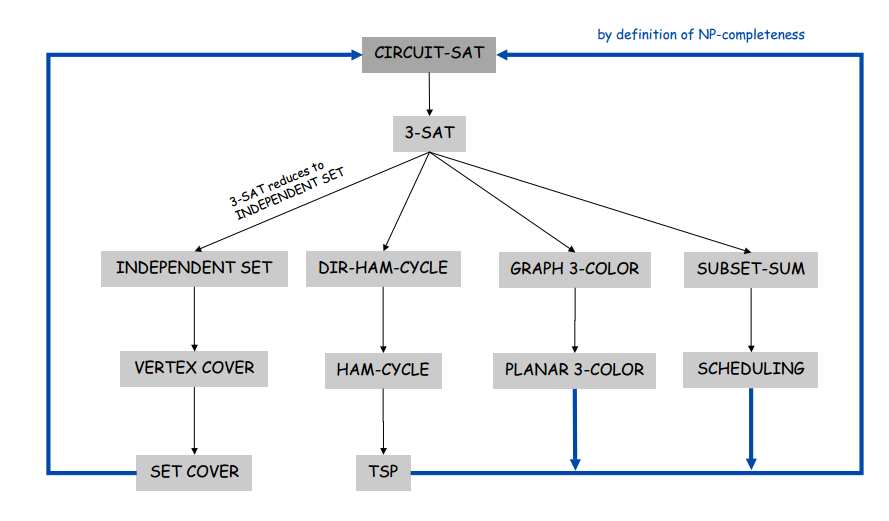
* Given n objects and a "knapsack."
* Item i has value vi > 0 and weighs wi > 0.
* Knapsack can carry weight up to W.
* Goal: fill knapsack so as to maximize total value.

#### Constraint Satisfaction Problems

**3-SAT**

Given a set of clauses C1,.....,Ck, each of length 3, over a set of variables

X={x1,....,xn}, does there exist a satisfying truth assignment?

**Packing problems:**

SET-PACKING, INDEPENDENT SET

**Covering problems:**

SET-COVER, VERTEX-COVER

**Constraint satisfaction problems:**

SAT, 3-SAT

**Sequencing problems:**

HAMILTONIAN-CYCLE, TSP

**Partitioning problems:**

3D-MATCHING, 3-COLOR

**Numerical problems:**

SUBSET-SUM, KNAPSACK

# Övriga vettiga saker

* Bipartit – grafen kan delas in i 2 mängder, sådan att inga kanter går mellan två noder i samma mängd
* Adjency – listor/matris med varje nods grannar
* Brute force - testar alla fall, ofta 2^n
* Degree (graphs) – antalet kanter kopplade till en nod
* \sum_{i=m}^n i = \frac{(n+1-m)(n+m)}{2}
* T(n) = 1 + T(sqrt(n), Låt https://piazza.com/main/show_latex?m%20%3D%20log%28n%29%20%5CRightarrow%202%5Em%20%3D%20n,  
    
  Substituera n i formeln,  
    
  https://piazza.com/main/show_latex?%5CRightarrow%20T%282%5Em%29%20%3D%201%20+%20T%28%5Csqrt%7B2%5Em%7D%29%20%3D%201%20+%20T%282%5E%7Bm/2%7D%29.  
    
  Skapa https://piazza.com/main/show_latex?S%28m%29%20%3D%20T%282%5Em%29 så får du:  
    
  https://piazza.com/main/show_latex?S%28m%29%20%3D%201%20+%20S%28m/2%29  
    
  Lös på vanligt sätt:  
    
  https://piazza.com/main/show_latex?%5CRightarrow%20S%28m%29%20%3D%20O%28log%28m%29%29%20%5CRightarrow%20T%28n%29%20%3D%20O%28log%28log%28n%29%29%29, vilket då är svaret

https://lh4.googleusercontent.com/IYs0dIoBJ4n1O6o-kbr0IwD745cghJSx9-e8sH6bKBL5uhV0ovcvQcuBM6RbVYE5PMokBRYsKhFO_JuRobQR2Z-lwfshEQlmmfAAAxFY4xgWQULZyp9soAEpgw

#### Common Recurrence Relations

|  |  |  |
| --- | --- | --- |
| T(n) = T(n/2) + O(1) | Binary search | *O(log (n))* |
| T(n) = T(n-1) + O(1) | Sequential search | *O(n)* |
| T(n) = 2T(n/2) + O(1) | Tree traversal | *O(n)* |
| T(n) = T(n-1) + O(n) | Selection sort (other n2 sorts) | *O(n2)* |
| T(n) = 2T(n/2) + O(n) | Mergesort (average case Quicksort) | *O(n log (n))* |
| T(n) = T(n-1) + T(n-2) + O(1) | Fibonacci | *O(2n)* |
| T(n) = T(n) + O(1) |  | *O(log (log (n)))* |
| *T(n) = T(n) + O(n)* |  | O(n) |