1 Prim's Algorithm (CLRS §23.2)

MST-PRIM(G, w, r)

```
1  for u in V
2       u.key =
3       u.parent = NIL
4  r.key = 0
5  Q = V
6  while Q
7       u = EXTRACT-MIN(Q)
8       for v in Adj[u]
9            if v in Q and w(u, v) < v.key
10                v.parent = u
11                v.key = w(u, v)
```

Example

Vertex	A	B	C	D	E	F	G	H
Key								
Parent								

Vertex	A	B	C	D	E	F	G	H
Key								
Parent								

Vertex	A	B	C	D	E	F	G	H
Key								
Parent								

Vertex	A	B	C	D	E	F	G	H
Key								
Parent								

Vertex	A	B	C	D	E	F	G	H
Key								
Parent								

Vertex	A	B	C	D	E	F	G	H
Key								
Parent								

2 Single-Source Shortest Paths (SSSP)

Input:

Goal:

Remarks:

- We might only care about a shortest path between $s \in V$ and one $t \in V$ (as opposed to between s and all V). The running time of algorithms for this single pair variant is not asymptotically faster.
- We can also consider negative weights, and there are algorithms for this extension like Bellman-Ford. Running time is $\mathcal{O}(VE)$.
- The case of all weights being 1

2.1 Dijkstra's Algorithm (§24.3)

Example

```
1  for each
2       v.d =
3       v.p =
4  s.d =
5  Q = V
6  while
7       u = EXTRACT-MIN
8       for each v
9            if v.d >
10                v.d =
11                v.p =
```

Running time:

Let $\delta(u, v) =$

Correctness:
Theorem. For any $u \in V \setminus Q$, at any point in the algorithm's execution,

In particular, Dijkstra's algorithm terminates with

Proof sketch.

Lemma. For any $v \in V$ at any point in the algorithm's execution,

2.2 Special case for DAG

Example

3 P, NP, NP-completeness

All the algorithms we've seen in the course had polynomial running time. E.g.

Problems that can be solved in polynomial time form the

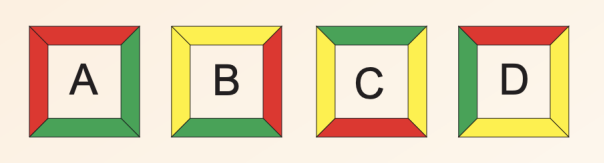
Do all problems have polynomial time algorithms?

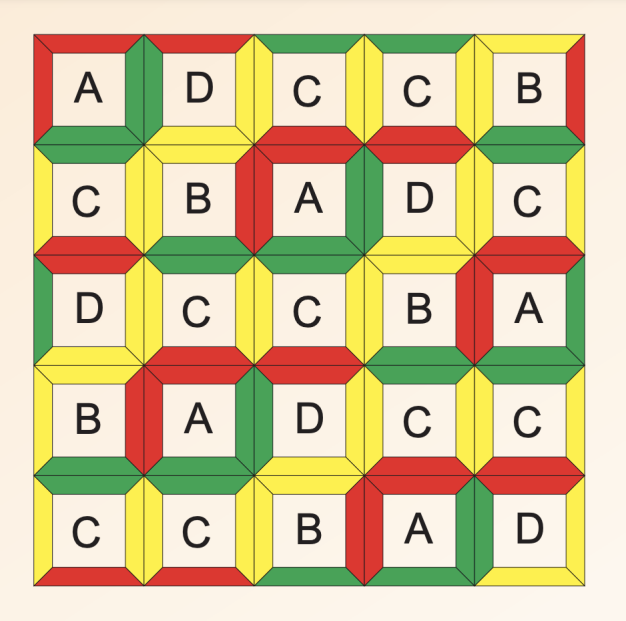
3.1 Uncomputability

Some problems cannot be solved

Example

A **Wang tile** is a square tile with colored edges.





Tiling problem. Given a finite set of Wang tiles, does it admit a valid tiling of the plane? (i.e. place the tiles so that touching edges of adjacent tiles have the same color)

Example Given code P for a program and a string x, does P halt when run on x?

Luckily, almost all problems encountered in real life are computable. However, many don't (seem to) have

How to deal with this?

1. Heuristics
2. Simplify problem, maybe assume something about input
3. Approximate
4. Buy more hardware (only gets you so far...)

3.2 Reductions

Definition A **vertex cover** of an undirected graph $G = (V, E)$ is

Definition An **independent set** of an undirected graph $G = (V, E)$ is

Remark.

Claim. VC $\leq$ IS; i.e., there is a *reduction* from VC to IS.

Proof.

Definition

Example

Example *3-SAT*

3.3 Complexity classes

Definition The **class NP** consists of YES/NO problems (*decision problems*) for which there is a polynomial time

Example

Definition A problem $A \in$ NP is called **NP-complete** if

Theorem. (Cook-Levin 1971) 3-SAT is NP-complete.
Corollary.

It is widely believed that $P \neq NP$ but proving it is beyond reach.

4 Approximating Vertex Cover (CLRS §35.1)

Even though VC is NP-complete (and likely requires exponential time), we can hope to find an approximate solution that is guaranteed to be not much worse than optimal.

Example

Approximate algorithm:

Theorem. This algorithm gives
Proof.