CS580 Algorithm Design, Analysis, and Implementation

Lecture notes, Jan 11, 2022
Wufei Ma

1 Introduction

Interval scheduling. The input is a set of jobs with start times and finish times. The goal is to find **maximum cardinality subset** of mutually compatible jobs. ($O(n \log n)$ greedy algorithm.)

Weighted interval scheduling. The input is a set of jobs with start times, finish times, and weights. The goal is to find **maximum weight** subset of mutually compatible jobs. ($O(n \log n)$ dynamic programming algorithm.)

Bipartite matching. The input is a bipartite graph. The goal is to find **maximum cardinality** matching. See Figure 1. ($O(n^k)$ max-flow based algorithm.)

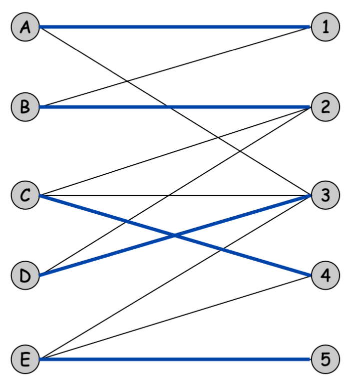

Figure 1: Bipartite matching.

Independent set. The input is a graph. The goal is to find **maximum cardinality** independent set. See Figure 2. (NP complete.)

2 Review

Desirable scaling property. When the input size doubles, the algorithm only slow down by some constant factor C. More formally, there exists constant $c > 0$ and $d > 0$ such that on every input of size N, its running time is bounded by cN^d steps.

Polynomial time. An algorithm is poly-time if the scaling property holds. An algorithm is **efficient** if its running time is polynomial.

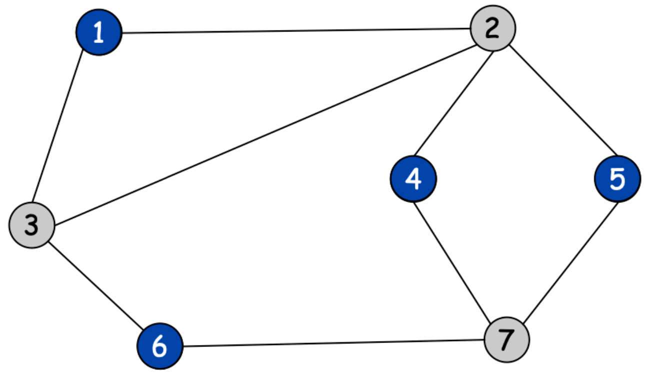

Figure 2: Independent set.

- Some exponential-time (or worse) algorithms are widely used because the worst-case instances seem to be rare.

Worst case running time. Obtain bound on **largest possible** running time of algorithm on input of a given size N.

Average case running time. Obtain bound on running time of algorithm on **random** input as a function of input size N.

Asymptotic order or growth.

- **Upper bound.** $T(n)$ is $O(f(n))$ if there exists constant $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$ we have $T(n) \leq c \cdot f(n)$.
- **Lower bound.** $T(n)$ is $\Omega(f(n))$ if there exists constant $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$ we have $T(n) \geq c \cdot f(n)$.
- **Tight bound.** $T(n)$ is $\Theta(f(n))$ if $T(n)$ is both $O(f(n))$ and $\Omega(f(n))$.

Asymptotic bounds for some common functions.

- Log grows slower than polynomial: for every $x > 0$, $\log n = O(n^x)$.
- Exponential grows faster than polynomial: for every $r > 1$ and every $d > 0$, $n^d = O(r^n)$.