BFS head ← 1, tail ← 1, queue[1] ← s
mark s as "visited"
color[s] ← 0
while head ≤ tail do
  v ← queue[head], head ← head+1
  for all neighbours u of v do
    if u is "unvisited" then
      tail ← tail+1, queue[tail]=u
      mark u as "visited"
      color[u] ← 1 − color[v]
    else if color[u] = color[v] then
      print "not bipartite"; exit

$O(n+m)$
if only if no odd cycle

Cycle Detect
for every $v \in V$, visited[v] ← false
for every $v \in V$ do
  if visited[v] = false then DFS(v)
print (no cycle)

DFS visited ← true
for every neighbor u of v do
  if visited[u] = false then
    par[u] ← v, DFS(u)
  else if u ≠ par[v] then
    print ("cycle", v), w ← v
    while w ≠ u do
      w ← par[w], print(w)
    exit

Topological ordering $O(n+m)$
Initilize d array over V, so d[v] is the in-degree of v in G
top ← 0, for every $v \in V$ do:
  if d[v] = 0 then top ← top+1, stack[top] ← v
unique ← true, sorted ← 0
while sorted < n do
  if top = 0 then return "none"
  if top ≥ 2 then unique ← false
  u ← stack[top], top ← top−1, sorted ← sorted+1
  for every outgoing edge (u, v) of u do
    d[v] ← d[v] − 1
    if d[v] = 0 then top ← top+1, stack[top] ← v
if unique = true then return "unique" else return "multiple"

Interval scheduling $O(n \lg n)$
Offline Caching: Furthest-in-Future

priority queue	insert	extract_min	decrease_key
array	$O(1)$	$O(n)$	$O(1)$
sorted array	$O(n)$	$O(1)$	$O(n)$
heap	$O(\lg n)$	$O(\lg n)$	$O(\lg n)$

$S \setminus \{u\} \cup \{v\}$; In any case there is a ··· contains

L ← empty max-heap, L.insert(−∞)
R ← empty min-heap, R.insert(∞)
for i ← 1 to n do
  if i is odd then
    if A[i] ≤ R.get_min() then
      L.insert(A[i])
    else R.insert(A[i]) L.insert(R.extract_min())
  else
    if A[i] ≥ L.get_max() then
      R.insert(A[i])
    else L.insert(A[i])
      R.insert(L.extract_max())
  b_i ← L.get_max()
Output $b_1, b_2, \cdots b_n$
$O(n \log n)$

Divide and Conquer
Closet-Pair $O(n \lg n)$
Convex-Hull $O(n \lg n)$
Matrix-Multiplication $T(n) = 7T(n/2) + O(n^2)$
Fibonacci DP: $O(n)$ D&C: $O(\lg n)^*$

Local min l ← 1, r ← n $O(\log(n))$
while l < r do ① m ← $\lfloor \frac{l+r}{2} \rfloor$
② if A[m] < A[m+1] then r ← m else l ← m+1
return l

majority_fare(l, r)
if l = r then return l
m ← $\lfloor \frac{l+r}{2} \rfloor$
a ← majority(l, m)
b ← majority(m+1, r)
if a ≠ ⊥ and check(l, r, a) then return a
if b ≠ ⊥ and check(l, r, b) then return b
return ⊥

check(l, r, t)
count ← 0
for i ← l to r do
  if same(i, t) then count ← count+1
return true if count > (r−l+1)/2
  false otherwise
$O(n \log n)$

Weighted interval scheduling $opt[i] = \max\{opt[i-1], w_i + opt[p_i]\}$, p_i is the largest j that $f_j \le s_i$ $O(n \lg n)$

subset sum/knapsack
$$opt[i, W'] = \begin{cases} 0 & i=0 \\ opt[i-1, W'] & i>0, w_i > W' \\ \max\begin{cases} opt[i-1, W'] \\ opt[i-1, W'-w_i] + \overset{v_i}{w_i} \end{cases} & i>0, w_i \le W' \end{cases}$$

LCS
$$opt[i,j] = \begin{cases} opt[i-1, j-1] + \overset{\to 2}{1} & \text{if } A[i] = B[j] \ \nwarrow \\ \max\begin{cases} opt[i-1, j] & \uparrow \\ opt[i, j-1] & \leftarrow \end{cases} & \text{if } A[i] \ne B[j] \end{cases}$$
(opt[i−1, j−1]+1) change letter
Space: $O(m+n)$
Time: $O(nm)$

Shortest in DAG
$$f[i] = \begin{cases} 0 & i=1 \\ \min_{j:(j,i)\in E}\{f(j) + w(j,i)\} & i = 2, 3, \dots n \end{cases}$$

Matrix chain multiply
$$opt[i,j] = \begin{cases} 0 & i=j \\ \min_{k: i \le k < j}(opt[i,k] + opt[k+1, j] + r_i c_k c_j) & i<j \end{cases}$$

optimum binary search
$$opt[i,j] = \begin{cases} 0 & \text{if } i = j+1 \\ \min_{k: i \text{ to } j}(opt[i, k-1] + opt[k+1, j]) + \sum_{l=i}^{j} f_l & \text{if } i \le j \end{cases}$$

Kruskal: union-find
Prim: priority queue

	extract_min	decrease_key	overall
heap	$O(\lg n)$	$O(\lg n)$	$O(m \lg n)$
fibonacci heap	$O(\lg n)$	$O(1)$	$O(n \lg n + m)$

$$BF: f^l(v) = \begin{cases} 0 & l=0, v=s \\ \infty & l=0, v \ne s \\ \min\begin{cases} f^{l-1}[v] \\ \min_{u:(u,v)\in E}(f^{l-1}[u] + w(u,v)) \end{cases} & l > 0 \end{cases}$$

$$FW \quad f^k[i,j] = \begin{cases} w(i,j) & k=0 \\ \min\begin{cases} f^{k-1}[i,j] \\ f^{k-1}[i,k] + f^{k-1}[k,j] \end{cases} & k = 1, 2, \dots n \end{cases}$$

DP	DAG	R	SS	$O(n+m)$
Dijkstra	U/D	$R_{\ge 0}$	SS	$O(n \lg n + m)$
Bellman-Ford	U/D	R	SS	$O(nm)$
Floyd-Warshall	U/D	R	AP	$O(n^3)$

$P = NP = Co\text{-}NP$ | $NP = Co\text{-}NP$ (P) | NP ($P = NP \cap Co\text{-}NP$) $Co\text{-}NP$ | NP ($NP \cap Co\text{-}NP$, P) $Co\text{-}NP$

NP-C $\begin{cases} X \in NP \\ Y \le_p X \text{ for every } Y \in NP \end{cases}$

$x_5 = x_1 \vee x_2 \Leftrightarrow$

$(x_1 \vee x_2 \vee \neg x_5) \wedge$
$(x_1 \vee \neg x_2 \vee x_5) \wedge$
$(\neg x_1 \vee x_2 \vee x_5) \wedge$
$(\neg x_1 \vee \neg x_2 \vee x_5)$

x_1	x_2	x_5	$x_5 \Leftrightarrow x_1 \vee x_2$	
0	0	0	1	
0	0	1	0	•
0	1	0	0	•
0	1	1	1	
1	0	0	0	•
1	0	1	1	
1	1	0	0	•
1	1	1	1	

longest increasing sub-sequence

$f[i] = \max_{j<i: A[j]<A[i]} f[j] + 1$

maximum-weight independent set

$opt[j] = \max \begin{cases} \sum_{i \text{ childs of } j} opt[i] \\ \sum_{i \text{ grandchilds of } j} opt[i] + w_j \end{cases}$

(post-order traversal)

palindrome $opt[i,j]$ length of longest palindrome subseq of $A[i..j]$

$$opt[i,j] = \begin{cases} 0 & j = i-1 \\ 1 & j = i \\ opt[i+1, j-1] + 2 & j \ge i+1 \text{ and } A[i] = A[j] \\ \max\{opt[i,j-1], opt[i+1,j]\} & j \ge i+1 \text{ and } A[i] \ne A[j] \end{cases}$$

compute in non-decreasing order of $j - i$

DFS odd cycle (bipartite)

for every $u \in V$ do $color[u] \leftarrow -1$
for every $u \in V$ do
  if $color[u] = -1$ then $color[u] \leftarrow 0$, $DFS(u)$

$DFS(u)$
  for every edge (u,v) adjacent to u do
    if $color[v] = -1$ then (not visited)
      $color[v] \leftarrow 1 - color[u]$, $parent[v] \leftarrow u$, $DFS(v)$
    else if $color[u] = color[v]$ then
      while $u \ne v$ do
        $print(u)$, $u \leftarrow parent[u]$
      $print[u]$; exit

DAG, $d[v]$: length of shortest path from s to v, $\pi[v]$ is vertex before v
check s to t is unique or not

$unique[s] \leftarrow true$
Sort vertices in $V \setminus \{s\}$ in non-decreasing order of d values
for every $v \in V \setminus \{s\}$ in the order do
  $unique[v] \leftarrow unique[\pi[v]]$
  for every incoming edge (u,v) of v do
    if $u \ne \pi[v]$ and $d[v] = d[u] + w(u,v)$ then $unique[v] \leftarrow false$
return $unique[t]$

$O(n \lg n + m)$