Big O notation: properties

Shanghai Jiao Tong University

Reflexivity. f is $O(f)$.

Constants. If f is $O(g)$ and $c > 0$, then $c\,f$ is $O(g)$.

Products. If f_1 is $O(g_1)$ and f_2 is $O(g_2)$, then $f_1 f_2$ is $O(g_1 g_2)$.

Proof.

- $\exists c_1 > 0$ and $n_1 \geq 0$ such that $0 \leq f_1(n) \leq c_1 \cdot g_1(n)$ for all $n \geq n_1$.
- $\exists c_2 > 0$ and $n_2 \geq 0$ such that $0 \leq f_2(n) \leq c_2 \cdot g_2(n)$ for all $n \geq n_2$.
- Then, $0 \leq f_1(n) \cdot f_2(n) \leq c_1 \cdot c_2 \cdot g_1(n) \cdot g_2(n)$ for all $n \geq \max\{n_1, n_2\}$.

Sums. If f_1 is $O(g_1)$ and f_2 is $O(g_2)$, then $f_1 + f_2$ is $O(\max\{g_1, g_2\})$.

Transitivity. If f is $O(g)$ and g is $O(h)$, then f is $O(h)$.

Ex. $f(n) = 5n^3 + 3n^2 + n + 1234$ is $O(n^3)$.

Asymptotic bounds and limits

Proposition

If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$ *for some constant* $0 < c < \infty$ *then* $f(n)$ *is* $\Theta(g(n))$.

Proof.

By definition of the limit, for any $\varepsilon > 0$, there exists n_0 such that

$$c - \varepsilon \leq \frac{f(n)}{g(n)} \leq c + \varepsilon$$

for all $n \geq n_0$.

Choose $\varepsilon = 1/2c > 0$.

Multiplying by $g(n)$ yields $1/2c \cdot g(n) \leq f(n) \leq 3/2c \cdot g(n)$ for all $n \geq n_0$.

Thus, $f(n)$ is $\Theta(g(n))$ by definition, with $c_1 = 1/2c$ and $c_2 = 3/2c$.

Euclid's Algorithm for Greatest Common Divisor

Q: Given two integers x and y, how to find their greatest common divisor ($gcd(x, y)$)?

Euclid's rule

If x and y are positive integers with $x \geq y$, then $gcd(x, y) = gcd(x \pmod y, y)$.

Proof:

It is enough to show the rule $gcd(x, y) = gcd(x - y, y)$. Result can be derived by repeatedly subtracting y from x.

Euclid's Algorithm for Greatest Common Divisor

SHANGHAI JIAO TONG UNIVERSITY

```
EUCLID(x, y)
Two integers x and y with x ≥ y;
if y = 0 then return x;
return(EUCLID(y, x mod y));
```

Lemma

If $a \geq b \geq 0$, *then* $a \mod b < a/2$

Proof:

- if $b \leq a/2$, $a \mod b < b \leq a/2$;
- if $b > a/2$, $a \mod b = a - b < a/2$.

An Extension of Euclid's Algorithm

Q: Suppose someone claims that d is the greatest common divisor of x and y, how can we check this?

It is not enough to verify that d divides both x and y...

Lemma

If d divides both x and y, and $d = ax + by$ for some integers a and b, then necessarily $d = gcd(x, y)$.

Proof:

$d \le gcd(x, y)$, obviously;

$d \ge gcd(x, y)$, since $gcd(x, y)$ can divide x and y, it must also divide $ax + by = d$.

Modular Inverse

Lemma

If $gcd(a, N) > 1$, then $ax \not\equiv 1 \mod N$.

Proof:

$ax \mod N = ax + kN$, then $gcd(a, N)$ divides $ax \mod N$

If $gcd(a, N) = 1$, then extended Euclid algorithm gives us integers x and y such that $ax + Ny = 1$, which means $ax \equiv 1 \mod N$. Thus x is a's sought inverse.

Fermat's Little Theorem

Theorem

If p is a prime, then for every $1 \leq a < p$,

$$a^{p-1} \equiv 1 \pmod{p}$$

Proof:

Let $S = \{1, 2, \ldots, p-1\}$, then multiplying these numbers by $a \pmod{p}$ is to permute them.

$a.i \pmod{p}$ are distinct for $i \in S$, and all the values are nonzero.

multiplying all numbers in each representation, then gives $(p-1)! \equiv a^{(p-1)}.(p-1)! \pmod{p}$, and thus

$$1 \equiv a^{(p-1)} \pmod{p}$$

Non-Carmichael Number

Lemma

If $a^{N-1} \not\equiv 1 \pmod{N}$ for some a relatively prime to N, then it must hold for at least half the choices of $a < N$.

Proof:

Fix some value of a for which $a^{N-1} \not\equiv 1(\mod N)$.

Assume some $b < N$ satisfies $b^{N-1} \equiv 1(\mod N)$, then

$$(a \cdot b)^{N-1} \equiv a^{N-1} \cdot b^{N-1} \equiv a^{N-1} \not\equiv 1(\mod N)$$

For $b \neq b'$, we have

$$a \cdot b \not\equiv a \cdot b' \mod N$$

The one-to-one function $b \mapsto a \cdot b(\mod N)$ shows that at least as many elements fail the test as pass it.

Proof of the Property

Proof:

If the mapping $x \to x^e \mod N$ is invertible, it must be a bijection; hence statement 2 implies statement 1.

To prove statement 2, observe that e is invertible modulo $(p-1)(q-1)$ because it is relatively prime to this number.

To show that $(x^e)^d \equiv x \mod N$: Since $ed \equiv 1 \mod (p-1)(q-1)$, can write $ed = 1 + k(p-1)(q-1)$ for some k.

Then

$$(x^e)^d - x = x^{ed} - x = x^{1+k(p-1)(q-1)} - x$$

$x^{1+k(p-1)(q-1)} - x$ is divisible by p (since $x^{p-1} \equiv 1 \mod p$) and likewise by q. Since p and q are primes, this expression must be divisible by $N = pq$.

The Proof of the Theorem

Proof:

Assume that n is a power of b.

The size of the subproblems decreases by a factor of b with each level of recursion, and therefore reaches the base case after $\log_b n$ levels - the the height of the recursion tree.

Its branching factor is a, so the k-th level of the tree is made up of a^k subproblems, each of size n/b^k.

$$a^k \times O(\frac{n}{b^k})^d = O(n^d) \times (\frac{a}{b^d})^k$$

k goes from 0 to $\log_b n$, these numbers form a geometric series with ratio a/b^d, comes down to three cases.

The Proof of the Theorem

The ratio is less than 1.

Then the series is decreasing, and its sum is just given by its first term, $O(n^d)$.

The ratio is greater than 1.

The series is increasing and its sum is given by its last term, $O(n^{\log_b a})$

The ratio is exactly 1.

In this case all $O(\log n)$ terms of the series are equal to $O(n^d)$.

The Efficiency Analysis

SHANGHAI JIAO TONG UNIVERSITY

v is good if it lies within the $25th$ to $75th$ percentile of the array that it is chosen from.

A randomly chosen v has a 50% chance of being good.

Lemma

On average a fair coin needs to be tossed two times before a heads is seen.

Proof:

Let E be the expected number of tosses before heads is seen.

$$E = 1 + \frac{1}{2}E$$

Therefore, $E = 2$.

Proposition

The sort-and-count algorithm counts the number of inversions in a permutation of size n in $O(n \log n)$ time.

Proof.

$$T(n) = 2 \cdot T(\lceil n/2 \rceil) + \Theta(n)$$

Interpolation resolved

Lemma

The columns of matrix M are orthogonal to each other.

Proof.

- Take the inner product of of any columns j and k of matrix M,

$$1 + \omega^{j-k} + \omega^{2(j-k)} + \ldots + \omega^{(n-1)(j-k)}$$

  This is a geometric series with first term 1, last term $\omega^{(n-1)(j-k)}$, and ratio ω^{j-k}.
- Therefore, if $j \neq k$, it evaluates to

$$\frac{1 - \omega^{n(j-k)}}{1 - \omega^{(j-k)}} = 0$$

- If $j = k$, then it evaluates to n.

Proof of Correctness

For each node $u \in S$, where S is the set of vertex with the $dist$ being set.

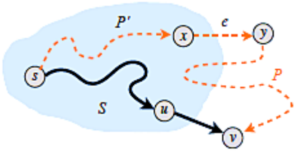
Proof. [by induction on $|S|$]

Base case: $|S| = 1$ is easy since $S = \{s\}$ and $dist[s] = 0$.

Inductive hypothesis: Assume true for $|S| \geq 1$.

- Let v be next node added to S, and let (u, v) be the final edge.
- A shortest $s \rightsquigarrow u$ path plus (u, v) is an $s \rightsquigarrow v$ path of length $\pi(v)$.
- Consider any other $s \rightsquigarrow v$ path P. We show that it is no shorter than $\pi(v)$.
- Let $e = (x, y)$ be the first edge in P that leaves S, and let P' be the subpath from s to x.
- The length of P is already $\geq \pi(v)$ as soon as it reaches y:

$$\begin{aligned} l(P) &\geq l(P') + \ell_e \\ &\geq dist[x] + \ell_e \\ &\geq \pi(y) \geq \pi(v). \end{aligned}$$



Proof of the Cut Property

Proof:

Edges X are part of some MST T; if the new edge e also happens to be part of T, then there is nothing to prove.

So assume e is not in T. We will construct a different MST T' containing $X \cup \{e\}$ by altering T slightly, changing just one of its edges.

Add edge e to T. Since T is connected, it already has a path between the endpoints of e, so adding e creates a cycle.

This cycle must also have some other edge e' across the cut $(S, V \backslash S)$. If we now remove e'

$$T' = T \cup \{e\} \backslash \{e'\}$$

which we will show to be a tree.

T' is connected by Lemma (1), since e' is a cycle edge. And it has the same number of edges as T; so by Lemma (2) and Lemma (3), it is also a tree.

Proof of the Cut Property

Proof:

T' is a minimum spanning tree, since

$$weight(T') = weight(T) + w(e) - w(e')$$

Both e and e' cross between S and $V \backslash S$, and e is the lightest edge of this type. Therefore $w(e) \leq w(e')$, and

$$weight(T') \leq weight(T)$$

Since T is an MST, it must be the case that $weight(T') = weight(T)$ and that T' is also an MST.

Performance Ratio

Lemma

Suppose B contains n elements and that the optimal cover consists of OPT sets. Then the greedy algorithm will use at most $\ln n \cdot OPT$ sets.

Proof.

Let n_t be the number of elements still not covered after t iterations of the greedy algorithm (so $n_0 = n$).

Since these remaining elements are covered by the optimal OPT sets, there must be some set with at least n_t/OPT of them.

Therefore, the greedy strategy will ensure that

$$n_{t+1} \leq n_t - \frac{n_t}{OPT} = n_t(1 - \frac{1}{OPT})$$

which by repeated application implies

$$n_t \leq n_0(1 - \frac{1}{OPT})^t$$

Properties of any optimal solution (for U.S. coin denominations)

Property. Number of pennies ≤ 4.

Proof. Replace 5 pennies with 1 nickel.

Property. Number of nickels ≤ 1.

Property. Number of quarters ≤ 3.

Property. Number of nickels + number of dimes ≤ 2.

Proof.

- Recall: ≤ 1 nickel.
- Replace 3 dimes and 0 nickels with 1 quarter and 1 nickel;
- Replace 2 dimes and 1 nickel with 1 quarter.

A rather formal proof

SHANGHAI JIAO TONG UNIVERSITY

Proof. by induction on amount to be paid x

Consider optimal way to change $c_k \leq x \leq c_{k+1}$: greedy takes coin k.

Claim that any optimal solution must take coin k.

- if not, it needs enough coins of type $c_1, \ldots, c_{k-1}$ to add up to x.
- table below indicates no optimal solution can do this

Problem reduces to coin-changing $x - c_k$ cents, which, by induction, is optimally solved by cashier's algorithm.

k	c_k	all optimal solutions must satisfy	max value of $c_1, c_2, \ldots c_{k-1}$ in any optimal solution
1	1	$P \leq 4$	none
2	5	$N \leq 1$	4
3	10	$N + D \leq 2$	$4 + 5 = 9$
4	25	$Q \leq 3$	$20 + 4 = 24$
5	100	no limit	$75 + 24 = 99$

Properties

Lemma (1)

For any non-root x, $\texttt{rank}(x) < \texttt{rank}(\pi(x))$.

Proof Sketch:

By design, the rank of a node is exactly the height of the subtree rooted at that node. This means, for instance, that as you move up a path toward a root node, the rank values along the way are strictly increasing.

Properties

Lemma (2)

Any root node of `rank` k has least 2^k nodes in its tree.

Proof Sketch:

A root node with rank k is created by the merger of two trees with roots of rank $k-1$. By induction to get the results.

Properties

Lemma (3)

If there are n elements overall, there can be at most $n/2^k$ nodes of $\texttt{rank}\ k$.

Proof Sketch:

A node of rank k has at least 2^k descendants.

Any internal node was once a root, and neither its rank nor its set of descendants has changed since then.

Different rank-k nodes cannot have common descendants. Any element has at most one ancestor of rank k.

Task 1 in the Origin

SHANGHAI JIAO TONG UNIVERSITY

Lemma

The origin is optimal if and only if all $c_i \leq 0$.

Proof.

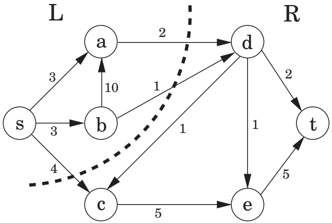
If all $c_i \leq 0$, then considering the constraints $x \geq 0$, we can't hope for a better objective value.

Conversely, if some $c_i > 0$, then the origin is not optimal, since we can increase the objective function by raising x_i.

Cuts

A truly remarkable fact:

Not only does simplex correctly compute a maximum flow, but it also generates a short proof of the optimality of this flow!



An (s,t)-cut partitions the vertices into two disjoint groups L and R, such that $s \in L$ and $t \in R$. Its capacity is the total capacity of the edges from L to R, and as argued previously, is an upper bound on any flow:

Pick any flow f and any (s,t)-cut (L,R). Then $\texttt{size}(f) \leq \texttt{capacity}(L,R)$.

A Certificate of Optimality

Proof:

Suppose f is the final flow when the algorithm terminates.

We know that node t is no longer reachable from s in the residual network G^f.

Let L be the nodes that are reachable from s in G^f, and let $R = V \setminus L$ be the rest of the nodes.

We claim that $\texttt{size}(f) = \texttt{capacity}(L, R)$.

To see this, observe that by the way L is defined, any edge going from L to R must be at full capacity (in the current flow f), and any edge from R to L must have zero flow.

Therefore the net flow across (L, R) is exactly the capacity of the cut.

P $\neq$ NP

Theorem Proving

- Input: A mathematical statement φ and n.
- Problem: Find a proof of φ of length $\leq n$ if there is one.

A formal proof of a mathematical assertion is written out in excruciating detail, it can be checked mechanically, by an efficient algorithm and is therefore in NP.

So if P $=$ NP, there would be an efficient method to prove any theorem, thus eliminating the need for mathematicians!

Approximation Guarantee Factor

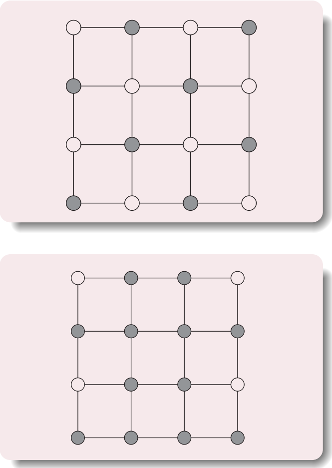
The Algorithm is a factor 2 approximation algorithm for the vertex cover problem.

Proof.

- No edge can be left uncovered by the set of vertices picked.
- Let M be the matching picked. As argued above,

$$|M| \leq OPT$$

- The approximation factor is at most $2 \cdot OPT$.



Proof for the Approximation Ratio 2

Let $x \in X$ be the point farthest from $\mu_1, \ldots, \mu_k$, and r be its distance to its closest center.

Then every point in X must be within distance r of its cluster center. By the triangle inequality, this means that every cluster has diameter at most $2r$.

We have identified $k + 1$ points $\{\mu_1, \mu_2, \ldots, \mu_k, x\}$ that are all at a distance at least r from each other.

Any partition into k clusters must put two of these points in the same cluster and must therefore have diameter at least r.

State-Flipping Algorithm: Proof of Correctness

Theorem

The state-flipping algorithm terminates with a stable configuration after at most $W = \sum_e |w_e|$ *iterations.*

Proof [Hint.] Consider measure of progress $\Phi(S) = \#$ satisfied nodes.

State-Flipping Algorithm: Proof of Correctness

SHANGHAI JIAO TONG UNIVERSITY

Theorem

The state-flipping algorithm terminates with a stable configuration after at most $W = \sum_e |w_e|$ *iterations.*

Proof. Consider measure of progress $\Phi(S) = \sum_{e \text{ good}} |w_e|$.

- Clearly $0 \le \Phi(S) \le W$.
- We show $\Phi(S)$ increase by at least 1 after each flip.

  When u flips state:
  - all good edges incident to u become bad
  - all bad edges incident to u become good
  - all other edges remain the same

$$\Phi(S') = \Phi(S) - \sum_{\substack{e : e = (u,v) \in E \\ e \text{ is bad}}} |w_e| + \sum_{\substack{e : e = (u,v) \in E \\ e \text{ is good}}} |w_e| \ge \Phi(S) + 1$$

Maximum Cut: Local Search Analysis

Theorem

Let (A, B) be a locally optimal cut and let (A^*, B^*) be an optimal cut. Then $w(A, B) \geq 1/2 \sum_e w_e \geq 1/2 w(A^*, B^*)$.

Proof.

- Local optimality implies that for all $u \in A : \sum_{v \in A} w_{uv} \leq \sum_{v \in B} w_{uv}$.
- Adding up all these inequalities yields: $2 \sum_{\{u,v\} \subseteq A} w_{uv} \leq \sum_{u \in A, v \in B} w_{uv} = w(A, B)$
- Similarly $2 \sum_{\{u,v\} \subseteq B} w_{uv} \leq \sum_{u \in A, v \in B} w_{uv} = w(A, B)$
- Now,

$$\sum_{e \in E} w_e = \underbrace{\sum_{\{u,v\} \subseteq A} w_{uv}}_{\leq \frac{1}{2} w(A,B)} + \underbrace{\sum_{u \in A, v \in B} w_{uv}}_{w(A,B)} + \underbrace{\sum_{\{u,v\} \subseteq A} w_{uv}}_{\leq \frac{1}{2} w(A,B)} \leq 2w(A, B)$$

Maximum Cut: Big Improvement Flips

Local search. Within a factor of 2 for MAX-CUT, but not polynomial time!

Big-improvement-flip algorithm. Only choose a node which, when flipped, increases the cut value by at least $\frac{2\varepsilon}{n} w(A, B)$

Claim

Upon termination, big-improvement-flip algorithm returns a cut (A, B) *such that* $(2+\varepsilon) w(A, B) \geq w\left(A^*, B^*\right)$

Proof idea. Add $\frac{2\varepsilon}{n} w(A, B)$ to each inequality in original proof.

Maximum Cut: Big Improvement Flips

Claim

Big-improvement-flip algorithm terminates after $O\left(\varepsilon^{-1} n \log W\right)$ *flips, where* $W = \sum_e w_e$.

Proof sketch.

Each flip improves cut value by at least a factor of $(1 + \varepsilon/n)$.

After n/ε iterations the cut value improves by a factor of 2.

- $(1 + 1/x)^x \geq 2$ for $x \geq 1$.

Cut value can be doubled at most $\log_2 W$ times.

Finding a Nash Equilibrium

Proof. Consider a set of $P_1, \dots, P_k$

- Let x_e denote the number of paths that use edge e.
- Let $\Phi(P_1, P_2, \dots P_k) = \sum_{e \in E} c_e \cdot H(x_e)$ be a potential function, where

$$H(0) = 0$$
$$H(k) = \sum_{i=1}^{k} \frac{1}{i}$$

- Since there are only finitely many sets of paths, it suffices to show that Φ strictly decreases in each step.

Finding a Nash Equilibrium

SHANGHAI JIAO TONG UNIVERSITY

Proof. [continued]

- Consider agent j switching from path P_j to path P'_j.
- Agent j switches because

$$\underbrace{\sum_{f \in P'_j - P_j} \frac{c_f}{x_f + 1}}_{\text{newly incurred cost}} < \underbrace{\sum_{e \in P_j - P'_j} \frac{c_e}{x_e}}_{\text{cost saved}}$$

- Φ increase by $\sum_{f \in P'_j - P_j} c_f \left[H\left(x_f + 1\right) - H\left(x_f\right) \right] = \sum_{f \in P'_j - P_j} \frac{c_f}{x_f + 1}$.
- Φ decrease by $\sum_{e \in P_j - P'_j} c_e \left[H\left(x_e\right) - H\left(x_e - 1\right) \right] = \sum_{e \in P_j - P'_j} \frac{c_c}{x_e}$
- Thus, net change in Φ is negative.

Bounding the Price of Stability

Lemma

Let $C(P_1, \ldots, P_k)$ denote the total cost of selecting paths $P_1, \ldots, P_k$. For any set of paths $P_1, \ldots, P_k$, we have

$$C(P_1, \ldots, P_k) \leq \Phi(P_1, \ldots, P_k) \leq H(k) \cdot C(P_1, \ldots, P_k)$$

Proof.

Let x_e denote the number of paths containing edge e.

- Let E^+ denote set of edges that belong to at least one of the paths. Then,

$$C(P_1, \ldots, P_k) = \sum_{e \in E^+} c_e \leq \underbrace{\sum_{e \in E^+} c_e H(x_e)}_{\Phi(P_1, \ldots, P_k)} \leq \sum_{e \in E^+} c_e H(k) = H(k) C(P_1, \ldots, P_k)$$

Bounding the Price of Stability

Theorem

There is a Nash equilibrium for which the total cost to all agents exceeds that of the social optimum by at most a factor of $H(k)$.

Proof.

- Let $(P_1^*, \ldots, P_k^*)$ denote a set of socially optimal paths.
- Run best-response dynamics algorithm starting from P^*.
- Since Φ is monotone decreasing $\Phi(P_1, \ldots, P_k) \leq \Phi(P_1^*, \ldots, P_k^*)$,

$$C(P_1, \ldots, P_k) \leq \Phi(P_1, \ldots, P_k) \leq \Phi(P_1^*, \ldots, P_k^*) \leq H(k) \cdot C(P_1^*, \ldots, P_k^*)$$