Note for RA

This note is based on the course AADS taught by UCPH.

- ***Why do we need randomized algorithms?*** (*Pros and Cons*)

Faster => But weaker guarantees

Simpler code but harder to analyze.

Sometimes it might be the only option, e.g., Big Data, Machine Learning, Security, etc.

- ***Classification of Randomized Algorithms***

Las Vegas & Monte Carlo.

Las Vegas algorithms always get a good answer but don't know how long it takes. => $RandQS(S)$.

Monte Carlo algorithms might give a wrong answer, but we have the trade-off between the running time and the probability of returning the correct solution. => $RandMinCut(G)$.

- ***QuickSort - Pseudocode***

The basic idea of the quicksort is sorting an array by comparing each element with the selected pivot in the iteration.

```
function QS(S={s1, ..., sn})
Assumes all elements in S are distinct.
  if |S| <= 1 then
    return list(S)
  else
    Pick pivot x in S
    L <- {y in S | y < x}
    R <- {y in S | y > x}
    return QS(L) + [x] + QS(R)
```

- ***QuickSort - Lemma 1***

In the basic algorithm, we don't specify how to pick the pivot. However, for any pivoting strategy, QS correctly sort the numbers.

Proof - By induction on n.

1. $n = 0, 1$ => Trivial
2. We assume it holds for up to $n - 1$ numbers.
3. Then by induction hypothesis $QS(L)$ and $QS(R)$ are sorted as their size are less or equal to $n - 1$.

Hence, $QS(L) + [x] + QS(R)$ is sorted.

① Why it can sort numbers?

当 n=0 时 → 能排

当 n=n-1 时 → 能排

当 n=n 时, RandQs[L] + [Si] + RandQs[R]

$\le n-1$ $\le n-1$

所以能排

- ***RandQS(S) - Pseudocode***

Randomized QuickSort Algorithm specifies the method of picking the pivot is to pick pivot $x \in S$ uniformly at random.

```
function RANDQS(S={s1, ..., sn})
Assumes all elements in S are distinct.
  if |S| <= 1 then
    return S
  else
    Pick pivot x in S, uniformly at random
    L <- {y in S | y < x}
    R <- {y in S | y > x}
    return RANDQS(L) + [x] + RANDQS(R)
```

- ***RandQS(S) - Analyse***

If lucky, everytime we pick the middle one to be the pivot during the iteration. Then, $|L| \le \frac{n}{2}$ and $|R| \le \frac{n}{2}$. The running time is, picking pivot + comparing,

$$
\begin{aligned}
T(n) &= O(n) + 2T(\frac{n}{2}) \\
&= O(n) + 2(O(\frac{n}{2}) + 2T(\frac{n}{4})) \\
&= O(n) + 2O(\frac{n}{2}) + 4O(\frac{n}{4}) + \ldots + nO(1) \\
&= O(n) + 2O(\frac{n}{2}) + \ldots + nO(1)
\end{aligned}
$$

As every term except $nO(1)$ is $O(\log n)$, hence the running time is $O(n \log n)$.

If we are unlucky, the running time should be $\Omega(n^2)$.

However, we show the interest on the average time.

From the pseudocode, we can know the running time of the algorithm is dominated by the number of comparisons.

What is the expected number of comparisons? => $\mathbb{E}[\#comparisons]$.

- ***Theorem 1***

$\mathbb{E}[\#comparisons] \in O(n \log n)$.

Proof

Let $[S_{(1)}, S_{(2)}, \ldots, S_{(n)}]$ is sorted by $RANDQS(S)$.

For $i < j$ let X_{ij} be the number of times that $S_{(i)}$ and $S_{(j)}$ are compared. We can observe that $X_{ij} \in \{0, 1\}$. That's because if one of them is selected to be the pivot, then it will be the only opportunity to get 1. Otherwise, they will never be compared to each other.

Then we get,

$$
\mathbb{E}[\#comparisons] = \mathbb{E}[\sum_{i<j} X_{ij}] = \sum_{i<j} \mathbb{E}[X_{ij}]
$$

Since $X_{ij} \in \{0, 1\}$ and it is an indicator variable for the event that $S_{(i)}$ and $S_{(j)}$ are compared. Let p_{ij} be the prabability of $S_{(i)}$ and $S_{(j)}$ are compared.

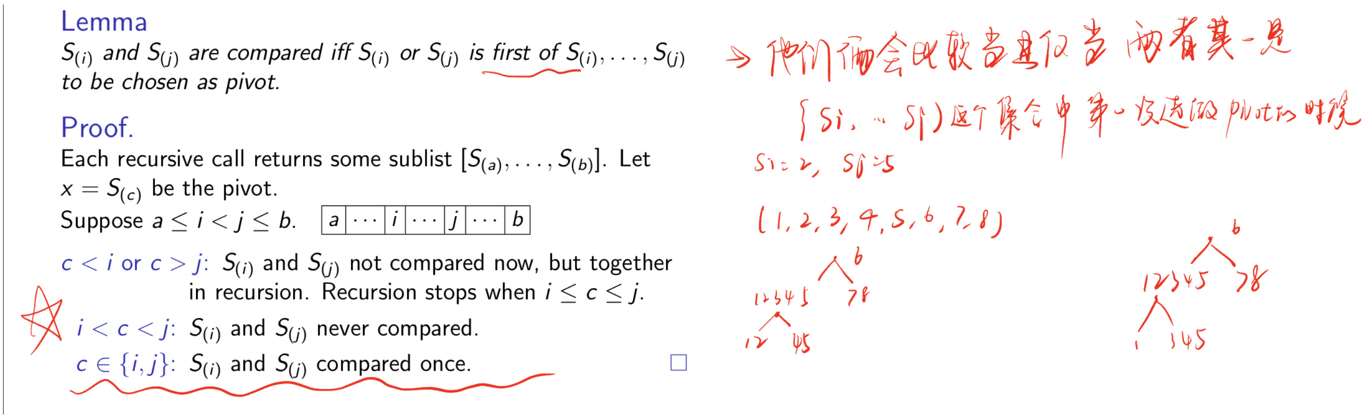
Then, $\mathbb{E}[X_{ij}] = (1 - p_{ij}) \cdot 0 + p_{ij} \cdot 1 = p_{ij}$.

Then we get $\sum_{i<j} \mathbb{E}[X_{ij}] = \sum_{i<j} p_{ij}$.

- ***Lemma 2***

$S_{(i)}$ and $S_{(j)}$ are compared if and only if $S_{(i)}$ or $S_{(j)}$ is first of the array $\{S_{(i)}, \dots, S_{(j)}\}$ to be chosen as pivot.

Proof



Lemma

$S_{(i)}$ and $S_{(j)}$ are compared iff $S_{(i)}$ or $S_{(j)}$ is first of $S_{(i)}, \dots, S_{(j)}$ to be chosen as pivot.

→ 他们俩会比较当且仅当 两者其一是 $\{S_i, \dots S_j\}$ 这个集合中第一次选做pivot的时候

Proof.

Each recursive call returns some sublist $[S_{(a)}, \dots, S_{(b)}]$. Let $x = S_{(c)}$ be the pivot.

Suppose $a \le i < j \le b$. | a | $\cdots$ | i | $\cdots$ | j | $\cdots$ | b |

$c < i$ or $c > j$: $S_{(i)}$ and $S_{(j)}$ not compared now, but together in recursion. Recursion stops when $i \le c \le j$.

$i < c < j$: $S_{(i)}$ and $S_{(j)}$ never compared.

$c \in \{i, j\}$: $S_{(i)}$ and $S_{(j)}$ compared once. □

$S_i = 2$, $S_j = 5$

$(1, 2, 3, 4, 5, 6, 7, 8)$

6

12345 78

12 45

6

12345 78

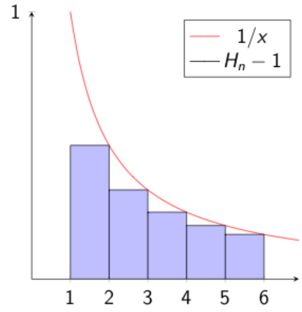
1 345

Thus, p_{ij} is the conditional probability of picking $S_{(i)}$ or $S_{(j)}$ given that the pivot is picked uniformly at random in $\{S_{(i)}, S_{(i+1)}, \dots, S_{(j)}\}$:

$$p_{ij} = \Pr[c \in \{i, j\} | c \in \{i, i+1, \dots, j\}, u.a.r.] = \frac{2}{j - i + 1}$$

Therefore, we get

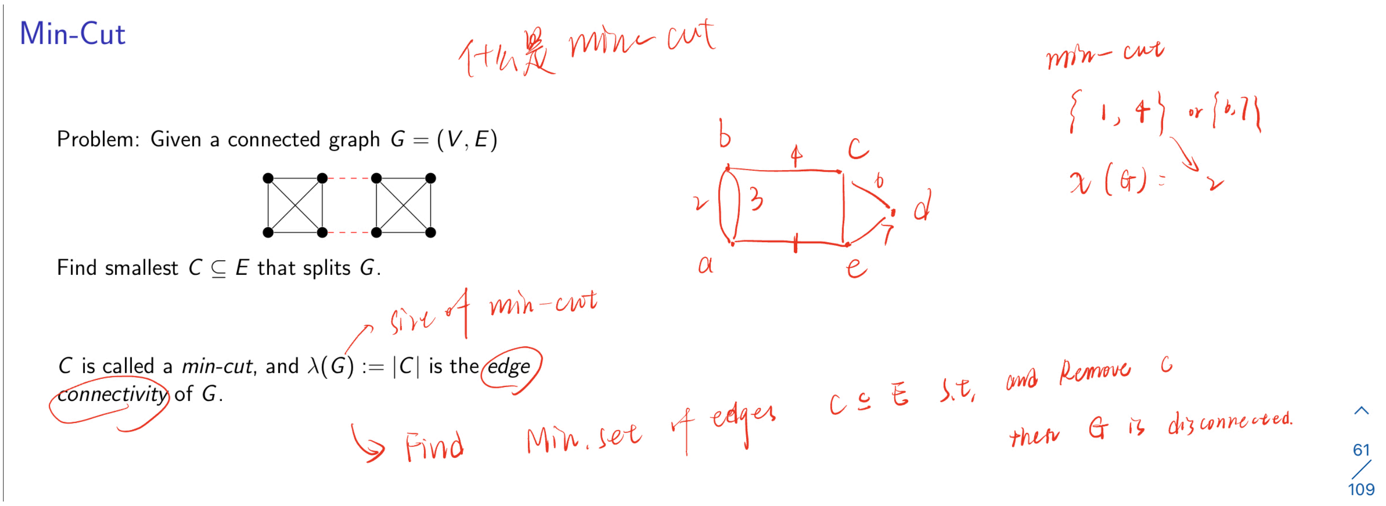
$$\begin{aligned}
\mathbb{E}[\#comparisons] &= \sum_{i<j} p_{ij} = \sum_{i<j} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1} \sum_{j=i}^{n} \frac{2}{j-i+1} \Leftarrow Extend \sum_{i<j} \\
Let\ k = j - i + 1,\ j_{\min} &= i + 1, k_{\min} = 2,\ j_{\max} = n, k_{\max} = n - i + 1 \\
&= \sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{2}{k} \\
&< \sum_{i=1}^{n} \sum_{k=2}^{n} \frac{2}{k} \Leftarrow add\ more \\
&= 2n \sum_{k=2}^{n} \frac{1}{k} \\
&= 2n((\sum_{k=1}^{n} \frac{1}{k}) - 1) \\
&= 2n(H_n - 1) \\
&\le 2n \int_1^n \frac{1}{x} dx \\
&= 2n \ln n = O(n \log n)
\end{aligned}$$



- ***RandQS - Summary***

When $|S| = n$, the $\mathbb{E}[\#comparisons] < 2nH_n \in O(n \log n)$ for any input.

- ***Min-cut***



$Min - cut$ problem. Find the smallest set C, which is the subset of the edge, that can make the original graph from connect to disconnect by removing the set.

C is called a min-cut, and $\lambda(G) = |C|$.

- ***RANDMINCUT - Pesudocode***

```
function RANDMINCUT(V, E)
while |V| > 2 and E is not empty do
  Pick e in E u.a.r.
  Contract e and remove self-loops.
return E
```

- ***RANDMINCUT - Lemma 1***

$RANDMINCUT(G)$ always returns a cut.

Proof => By induction on the number k of iterations of the loop ($k \le n-2$).

1. $k = 0$ is trivial.
2. Suppose that it is true for up to $k-1$ iterations.
3. First iteration constructs G' by contracting an edge from G and removing self-loops, and then, do at most $k-1$ further iterations starting from G' so by the induction hypothesis we return a cut in G'.

But every such cut is also a cut in G.

Randomized Min-Cut, Analysis

```
function RandMinCut(V, E)
    while |V| > 2 and E ≠ ∅ do
        Pick e ∈ E uniformly at random. ①
        Contract e and remove self-loops. ②
    return E
```

Lemma

RandMinCut(G) *always returns a cut.* ?

Proof.

Proof by induction on the number k of iterations of the loop (note $k \le n-2$). If $k = 0$ it is trivial, so suppose that it is true for up to $k-1$ iterations. The first iteration constructs graph G' by contracting an edge from G and removing self-loops, and then do at most $k-1$ futher iterations starting from G' so by the induction hypothesis we return a cut in G'. But every such cut is also a cut in G. □

归纳法
① k=0 易得
② k-1次正确
③ k次时，第一遍从G中拿个边 ☆
收缩 然后再做k-1次
∴返回仍然是cut

or 反证法不太会

- ***RANDMINCUT - Observation***

We observe that $RANDMINCUT(G)$ may return a cut of size $> \lambda(G)$.

Lemma

A specific min-cut C is returned if and only if no edge from C was contracted.

- ***RANDMINCUT - Theorem***

For any min-cut C, the probability that $RANDMINCUT(G)$ returns C is $\ge \frac{2}{n(n-1)}$.

Proof

The case that the algorithm can return the min-cut is all the contracted edges removed before are not in the C.

Randomized Min-Cut, Analysis

Theorem
For any min-cut C, the probability that $\textsc{RandMinCut}(G)$ *returns C is* $\geq \frac{2}{n(n-1)}$.

Let $e_1, \ldots, e_{n-2}$ be the contracted edges, let $G_0 = G$ and $G_i = G_{i-1}/e_i$.
Let $\mathcal{E}_i$ be the event that $e_i \notin C$.
C is returned iff $\mathcal{E}_1 \cap \cdots \cap \mathcal{E}_{n-2}$.

Goal: $\Pr[\mathcal{E}_1 \cap \cdots \cap \mathcal{E}_{n-2}] \geq \frac{2}{n(n-1)}$

- In words, $\mathcal{E}_i$ is the event that the ith edge contracted is not in C, i.e., the ith contraction does not destroy C.
- $\mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-2}$ is thus the event that C is not destroyed in any step of the algorithm.

Thus, our goal is $\Pr[\epsilon_1 \wedge \ldots \wedge \epsilon_{n-2}] \geq \frac{2}{n(n-1)}$.

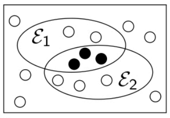
To know the probability of and operation.

Conditional Probabilities

Given events $\mathcal{E}_1, \mathcal{E}_2$, the *conditional probability* of $\mathcal{E}_2$ given $\mathcal{E}_1$ is defined as

$\mathcal{E}_1$在$\mathcal{E}_2$发生的条件下发生的概率

$$\Pr[\mathcal{E}_2 | \mathcal{E}_1] = \frac{\Pr[\mathcal{E}_1 \cap \mathcal{E}_2]}{\Pr[\mathcal{E}_1]}$$



$\mathcal{E}_1$和$\mathcal{E}_2$都发生

It follows that

$$\Pr[\mathcal{E}_1 \cap \mathcal{E}_2] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 | \mathcal{E}_1]$$

And in general for events $\mathcal{E}_1, \ldots, \mathcal{E}_k$

$$\Pr[\cap_{i=1}^{k} \mathcal{E}_i] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 | \mathcal{E}_1] \cdots \Pr[\mathcal{E}_k | \cap_{i=1}^{k-1} \mathcal{E}_i]$$

This is easy to prove by induction.

Then, the goal is converted to $\prod_{i=1}^{n-2} p_i$ where $p_i = \Pr[\epsilon_i | \epsilon_1 \wedge \ldots \wedge \epsilon_{i-1}]$. => 待求，返回最小割即前i次都没有选到最小割集合里的边。=> ***Stage 1***

---

1. We can know that $G_i = (V_i, E_i)$ has $n_i = n - i$ vertices as every time, the contraction operation will get rid of one vertex.
2. Contractions can not decrease the min-cut size, so $\lambda(G_i) \geq |C|$.
   - 至少等于，关于大于，因为G_i中的切同样也是G中的切，如果有小于的情况，则G的最小切就不是最小切。=> 同样也是上面一个观察的证明。
3. It follows that each vertex v of G_i has degree $d_i(v)$ at least $|C|$.

   If there is a $d_i(v)_{\min}$ in G_i such that $d_i(v)_{\min} < \lambda(G_i)$, it is contradicting the original min-cut in G_i is min-cut. => 挪掉一个点的度的边数，相当于把这个点挪出当前图，即图不再联通。Hence, $d_i(v)_{\min} \geq \lambda(G_i) \geq |C|$.
4. Summing up all degrees of G_i, with the usage of handshaking,

Randomized Min-Cut, Proof

最小

$G_i = (V_i, E_i)$ has $n_i = n - i$ vertices. (why?)
Contractions can not decrease the min-cut size, so $\lambda(G_i) \geq |C|$.
It follows that each vertex v of G_i has degree $d_i(v)$ at least $|C|$. (why?)
Summing up all degrees of G_i,

$$|E_i| = \frac{1}{2}\sum_{v \in V_i} d_i(v) \geq \frac{1}{2} n_i |C|.$$

边的个数 = 图的所有点的度的和 / 2

握手 handshaking

- Note that the edges incident to a vertex v form a cut and so $d_i(v) \geq \lambda(G_i) \geq |C|$.
- We use that each edge is counted twice in the sum $\sum_{v \in V_i} d_i(v)$.

$\min d_i v \geq |C|$

$\sum d_i(v) \geq n_i \min d_i v \geq n_i |C|$

Finally, we get $|E_i| \geq \frac{1}{2} n_i |C|$. => ***Stage 2***

Randomized Min-Cut, Proof

We have shown that $G_i = (V_i, E_i)$ has $n_i = n - i$ vertices and that $|E_i| \geq \frac{1}{2} n_i |C|$. We want to bound

$$p_i = \Pr[\underbrace{\text{uniformly random } e \in E_{i-1} \text{ is not in } C}_{A} \mid \underbrace{\cap_{j=1}^{i-1} \mathcal{E}_j}_{B}]$$

→ p_i = 在前 $i-1$ 次迭代不选 C 里的边的情况下第 i 次不选 C 的概率

The probability of picking an edge of C in the ith iteration, given that no edge of C has been picked in a previous iteration, is

$$1 - p_i = \Pr[\text{uniformly random } e \in E_{i-1} \text{ is in } C \mid \cap_{j=1}^{i-1} \mathcal{E}_j]$$

$$= \frac{|C|}{|E_{i-1}|} \leq \frac{|C|}{\frac{1}{2} n_{i-1} |C|} = \frac{2}{n_{i-1}} = \frac{2}{n - (i-1)}$$

$$\Rightarrow p_i \geq 1 - \frac{2}{n-i+1} = \frac{n-i-1}{n-i+1}$$

第二步是条件概率公式。n_{i-1} 是$i-1$次迭代剩下的点，1次少一个点，所以$n-(i-1)$。

Then, we can get $p_i \geq 1 - \frac{2}{n-i+1} = \frac{n-i-1}{n-i+1}$. => ***Stage 3***

$$\begin{aligned}
&\Pr[C \text{ returned}] \\
&= \prod_{i=1}^{n-2} p_i \quad \text{where } p_i = \Pr[\mathcal{E}_i | \mathcal{E}_1 \cap \cdots \cap \mathcal{E}_{i-1}] \\
&\geq \prod_{i=1}^{n-2} \frac{n-1-i}{n+1-i} \\
&= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{3}{5} \cdot \frac{2}{4} \cdot \frac{1}{3} \\
&= \frac{2}{n(n-1)}
\end{aligned}$$

Therefore, for a given min-cut, $\Pr[C \text{ } is \text{ } returned] \geq \frac{2}{n(n-1)}$.

Randomized Min-Cut, Summary

So for given min-cut C, $\Pr[C \text{ is returned}] \geq \frac{2}{n(n-1)}$.

Is this tight? I.e. do we have examples matching this bound? Yes! Consider the cycle C_n on n vertices. Every one of the $\binom{n}{2} = \frac{n(n-1)}{2}$ pairs of edges is a min-cut and all pairs are equally likely to be returned.

Is this probability good?

How can we improve it?

每个最小切被返回的概率是 $\frac{2}{n(n-1)}$

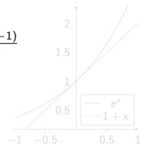
跑 $\frac{n(n-1)}{2}$ 次，则会返回最小切

次

Randomized Min-Cut, Tradeoff

Imagine calling RandMinCut(G) $t\frac{n(n-1)}{2}$ times and taking smallest cut returned.

$$\Pr[\text{not a min-cut}] \leq \left(1 - \frac{2}{n(n-1)}\right)^{t\frac{n(n-1)}{2}}$$



$$\leq \left(e^{-\frac{2}{n(n-1)}}\right)^{t\frac{n(n-1)}{2}}$$

(This uses that $1 + x \leq e^x$ for all $x \in \mathbb{R}$, see Proposition B.3.1)

$$= e^{-t}$$

$$\Pr[\text{min-cut}] \geq 1 - e^{-t}$$

$$= 1 - n^{-c} \qquad (\text{If we set } t = c \ln n)$$

Thus for any $c > 0$ if we repeat $c \cdot \frac{n(n-1)}{2} \cdot \ln n$ times, the probability of getting a min-cut is at least $1 - n^{-c}$. We call this *high probability of success.*

$(1-\frac{1}{x})^x$

$\leq (e^{-\frac{1}{x}})^x$

$= \frac{1}{e}$

- In each call to RandMinCut(G), the probability that a min-cut is not returned is at most $1 - \frac{2}{n(n-1)}$.
- Since the calls to RandMinCut(G) are independent, the probability that no min-cut is among the cuts returned is the product.
- $1 + x \leq e^x$
- Choosing e.g. $t = 21$ we reduce the error probability to around one in a billion.
- Choosing $t = c \ln n$ for constant c, we get a *high probability of success*, namely at least $1 - e^{-c \ln n} = 1 - 1/n^c$.
- We thus get a tradeoff between running time and probability of success.

- ***Tradeoff***

Above

- ***Simple Implementation***

In practice, using a "union-find" data structure the running time is as the following. => ***Polynomial*** time.

Randomized Min-Cut, Simple implementation

In practice, using a "Union-Find" data structure.

function RANDMINCUT(V, E)
  for $u \in V$ **do**
    MAKE-SET(u) $\Rightarrow O(n)$
  $C \leftarrow \emptyset$, $\pi \leftarrow$ a random permutation of E, $r \leftarrow |V|$
  for $uv \in E$ in the order π **do**
    $p_u \leftarrow$ FIND(u), $p_v \leftarrow$ FIND(v) $\Rightarrow O(m)$
    if $p_u \neq p_v$ **then**
      if $r > 2$ **then**
        $r \leftarrow r - 1$
        UNION(p_u, p_v) $\Rightarrow O(n)$ union take $O(m\alpha(n))$ time
      else
        $C \leftarrow C \cup \{uv\}$
  return C

disjoint sets

$\Rightarrow$ 所以在 polynomial 时间内运行

The running time for this is $\mathcal{O}(m\alpha(n))$. Running it $\mathcal{O}(n^2)$ times to get high probability takes $\mathcal{O}(n^2 m\alpha(n))$ time.

- ***Conversion***

L -> M 在结束前停止，那么返回的一定不是最佳答案。

M -> L 重复多次，一定会返回最佳结果。