正しく作るランダムケース

UV Study: ビアバッシュ！ 競技プログラミング LT 会

tsutaj

July 11, 2023

自己紹介

- Twitter: @tsutaj
- 最近は AHC に出たり ICPC OB/OG 会にいます
- 学生のときは大学有志で作問をよくやってました
  - 立命合宿、会津合宿、北大合宿、etc...
- 最近 PAST 上級〜エキスパート本を執筆しました
- 今回は競プロ作問に使えそうなテクを紹介します
  - 過去の発表 (参考): 競プロ作問を支える技術 Link
  - なけなしの作問ノウハウを少しずつ放流したい

1. はじめに
   - ランダムケースを使う場面
   - この発表における 「正しい」 ランダムケースとは

2. ランダムケースを作ってみよう
   - 順列をランダムに作る
   - N 個の中から K 個選ぶ
     - $K = 2$ だったら？
     - 確率的手法
     - 決定的手法
   - グラフをランダムに作る
     - グラフを作る
     - 木を作る

1. はじめに
   - ランダムケースを使う場面
   - この発表における「正しい」ランダムケースとは

2. ランダムケースを作ってみよう
   - 順列をランダムに作る
   - N 個の中から K 個選ぶ
     - $K = 2$ だったら？
     - 確率的手法
     - 決定的手法
   - グラフをランダムに作る
     - グラフを作る
     - 木を作る

ランダムケースを使う場面

ランダムにケースを作りたくなる場面はいくつかある

- 競プロで問題を解いていて、バグを特定したいとき
- 競プロのテストケースを作りたいとき
- 実行するごとに異なる出力結果にしたいとき
  - 人員の割り当て
  - 抽選

このスライドで 「正しい」 ランダムケースを作れるようになりましょう！

1. はじめに
   - ランダムケースを使う場面
   - この発表における 「正しい」 ランダムケースとは

2. ランダムケースを作ってみよう
   - 順列をランダムに作る
   - N 個の中から K 個選ぶ
     - $K = 2$ だったら？
     - 確率的手法
     - 決定的手法
   - グラフをランダムに作る
     - グラフを作る
     - 木を作る

この発表における「正しい」ランダムケースとは

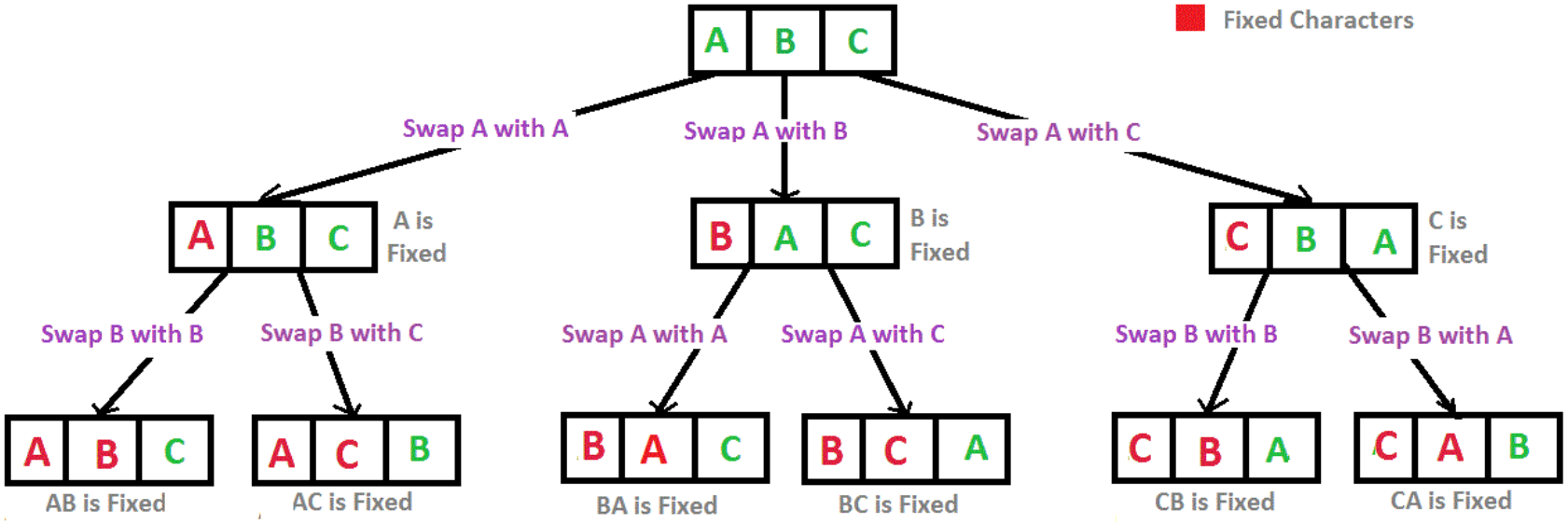
- この発表において、「正しいランダムケース」 = 偏りがない
  - アルゴリズムを間違えると、一様でなくなることも多々ある
- テストケースが偏ると、特定の要素がたくさん選ばれて困ることも
  - 例: そのシャッフル、本当にシャッフルですか？ 何気ない落とし穴にハマった話 Link
  - 同じペアが頻繁に選ばれてしまっている
- この発表では以下の関数の存在を認め、頻繁に使います
  - $\mathrm{rand}(a, b) := a$ 以上 b 未満の整数を一様ランダムに返す

1 はじめに
- ランダムケースを使う場面
- この発表における「正しい」ランダムケースとは

2 ランダムケースを作ってみよう
- 順列をランダムに作る
- N 個の中から K 個選ぶ
  - $K = 2$ だったら？
  - 確率的手法
  - 決定的手法
- グラフをランダムに作る
  - グラフを作る
  - 木を作る

順列をランダムに作る

- 長さ N の順列 P は $N!$ 通りある
- 一様ランダムに順列を作るには、$N!$ 通りの順列すべてが同じ確率で出ればよい
- 次ページでランダムに作る方法を紹介



Recursion Tree for Permutations of String "ABC"

Figure: 引用元: GeeksforGeeks Link

順列をランダムに作る

長さ N の順列 P を作るには、列 $A = (1, \ldots, N)$ をシャッフルすればよい

フィッシャー・イェーツのシャッフル Link

1: $P \leftarrow A$
2: **for** $i = 0, \ldots, N-2$:
3: $\quad j \leftarrow \mathrm{rand}(i, N)$
4: $\quad P_i$ と P_j を交換する
5: **end for**
6: **return** P

- 特徴: 計算量が $O(N)$ で、かつ偏りがない！
  - A で i 番目 (0-indexed) だった要素は、P で $N-i$ 通りの場所に入り得る
  - そのため、このアルゴリズムで得られる出力は $N!$ 通りある
    - 順列の総数に一致
    - 長さ N のすべての順列がこの方法で生成されうる

1 はじめに
- ランダムケースを使う場面
- この発表における「正しい」ランダムケースとは

2 ランダムケースを作ってみよう
- 順列をランダムに作る
- N 個の中から K 個選ぶ
  - $K = 2$ だったら？
  - 確率的手法
  - 決定的手法
- グラフをランダムに作る
  - グラフを作る
  - 木を作る

N 個の中から 2 個選ぶ方法

- $(0, 1, \ldots, N-1)$ の中から 2 個選ぶことを考える
- はじめに、よくある間違いを紹介
  - 私も実際にやってしまったことのある手法

N 個の中から 2 個選ぶ方法（間違い）

1: $a \leftarrow \mathrm{rand}(0, N-1)$
2: $b \leftarrow \mathrm{rand}(a+1, N)$
3: **return** (a, b)

- これは同様に確からしくならないので注意！
  - 具体例を出して考えてみましょう

N 個の中から 2 個選ぶ方法

N 個の中から 2 個選ぶ方法（間違い）

1: $a \leftarrow \mathrm{rand}(0, N-1)$
2: $b \leftarrow \mathrm{rand}(a+1, N)$
3: **return** (a, b)

- 偏っていることを確かめてみよう
  - $a = 0, b = 1$ となる確率: $\frac{1}{(N-1)^2}$
    - a が 0 になる確率も、b が 1 になる（条件付き）確率も $\frac{1}{N-1}$
  - $a = N-2, b = N-1$ となる確率: $\frac{1}{N-1}$
    - a が $N-2$ になる確率が $\frac{1}{N-1}$ で、そのとき b は必ず $N-1$ になる
  - 両者は本来同じ確率で出てほしいが、後者のほうが $N-1$ 倍出やすく、偏っている

N 個の中から 2 個選ぶ方法

では、どのようにすれば偏りなく取れるか？

N 個の中から 2 個選ぶ方法

```
repeat
    a ← rand(0, N)
    b ← rand(0, N)
until a ≠ b
```

- $a = b$ のケースを排除しつつ a, b をランダムに決めるのを繰り返すことで、一様ランダムにペアを決めることができる
- $a < b$ にしたい場合は、上記の方法で a, b を選んだ後に必要に応じて $\mathrm{swap}(a, b)$ すればよい

N 個の中から K 個選ぶ方法: expected $O(K)$

- $K \leq \frac{N}{2}$ のとき、次の方法で expected $O(K)$ を達成できる
  - $K \leq \frac{N}{2}$ より、要素を 1 個選ぶために必要な試行回数の期待値は 2 以下
  - S は unordered_set などで持っておくとよい

N 個の中から K 個選ぶ方法 ($K \leq \frac{N}{2}$ のとき)

```
S = ()
for i = 0, ..., K − 1 :
    repeat
        v ← rand(0, N)
    until v not in S
    S に v を追加する
end for
return S
```

- $K \geq \frac{N}{2}$ のときは、選ばない要素を上記の方法で決定することで expected $O(K)$ を達成できる
  - 補集合を考えている

N 個の中から K 個選ぶ方法: deterministic $O(K \log K)$

- 次の方法で deterministic $O(K \log K)$ を達成できる

N 個の中から K 個選ぶ方法

$t \leftarrow N - K + 1, d \leftarrow$ 長さ K の配列
for $i = 0, \ldots, K-1$:
    $r \leftarrow \mathrm{rand}(0, t+i)$
    if $r < t$: $d[i] \leftarrow r$
    else $d[i] \leftarrow d[r-t]$
    end if
end for
d を昇順ソートする
for $i = 0, \ldots, K-1$:
    $d[i] \leftarrow d[i] + i$
end for
return d

- このアルゴリズムのアイデアはこちらを参照のこと Link

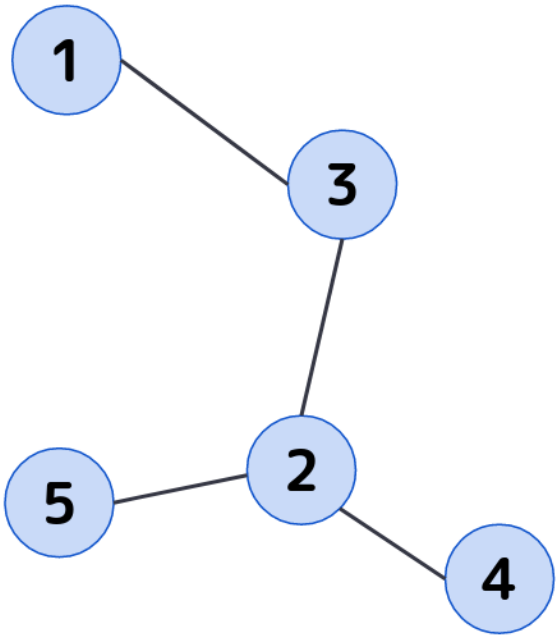
1 はじめに
- ランダムケースを使う場面
- この発表における「正しい」ランダムケースとは

2 ランダムケースを作ってみよう
- 順列をランダムに作る
- N 個の中から K 個選ぶ
  - $K = 2$ だったら？
  - 確率的手法
  - 決定的手法
- グラフをランダムに作る
  - グラフを作る
  - 木を作る

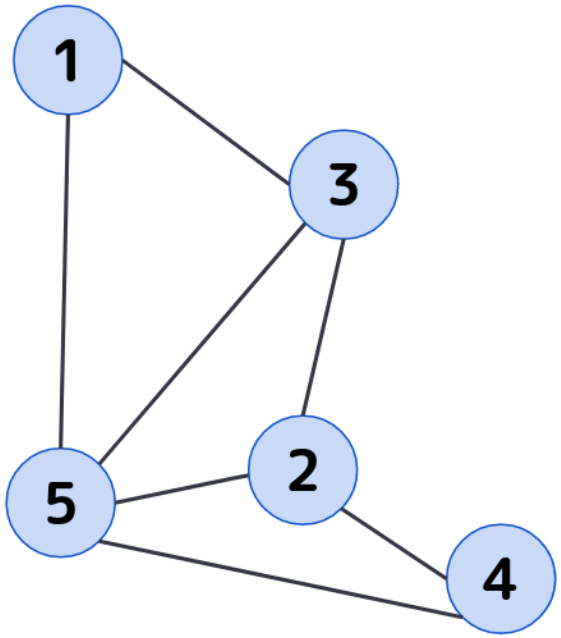
グラフを作る方法

頂点を区別した（頂点に番号がついた）グラフを作ることを考える

- ランダムグラフを作る
- 木を作る



木



グラフ

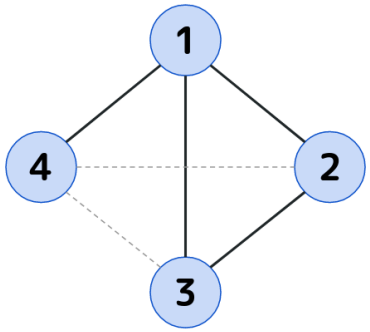
ランダムグラフを作る方法

- N 頂点 M 辺のランダムグラフを作ることを考える
  - ありえる辺の総数: $E = \frac{N(N-1)}{2}$

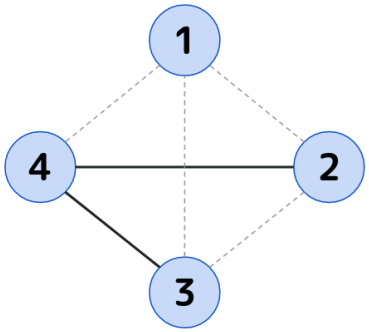
N 頂点 M 辺のランダムグラフを作る方法

先ほどの「N 個の中から K 個選ぶ方法」を応用

- $M < \frac{E}{2}$ のとき
  - 辺をランダムに M 個選び、N 頂点 M 辺からなるグラフ G を返す
- $M \geq \frac{E}{2}$ のとき
  - 辺をランダムに $E - M$ 個選ぶ
  - N 頂点 $E - M$ 辺からなるグラフ G を作る
  - G の補グラフ $\overline{G}$ を返す



グラフ



補グラフ
（辺の有無を逆にする）

ランダムグラフを作る方法（おまけ）

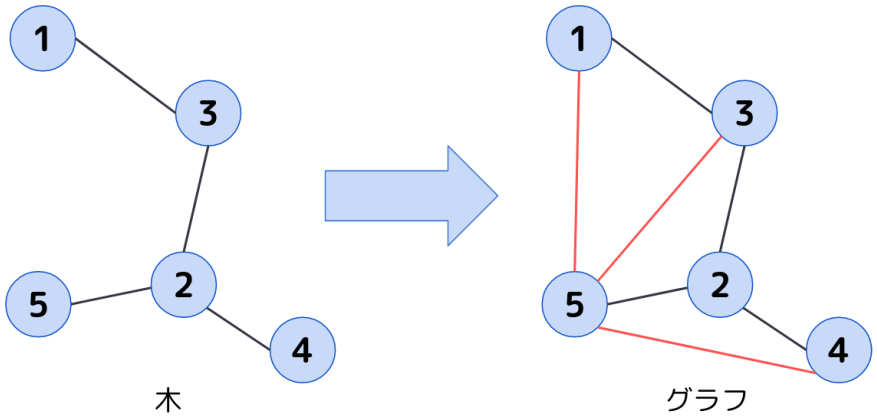
N 頂点 M 辺の「連結な」ランダムグラフを作る場合はどうするか？

よく見られる、連結なランダムグラフを作る手法

- N 頂点からなる木を作る
  - 木をランダムに作る方法は後述
- 残り $M - N + 1$ 辺をランダムに追加

しかし、厳密にはこれは一様ランダムでない

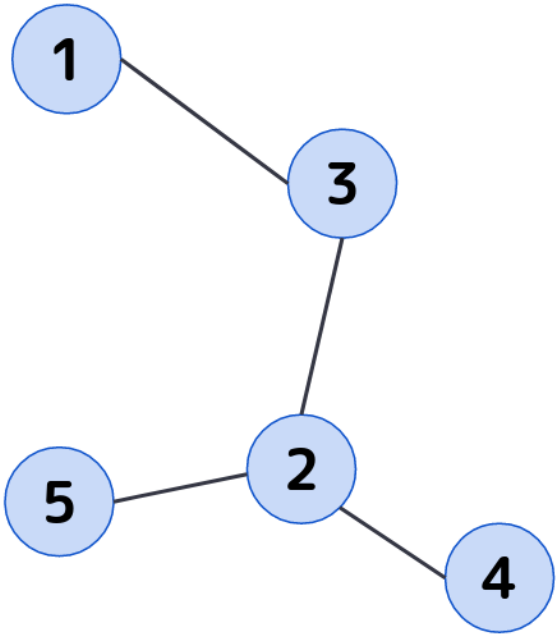
- 厳密に一様ランダム、かつ高速に作る方法があれば教えてください
- Codeforces での議論 Link



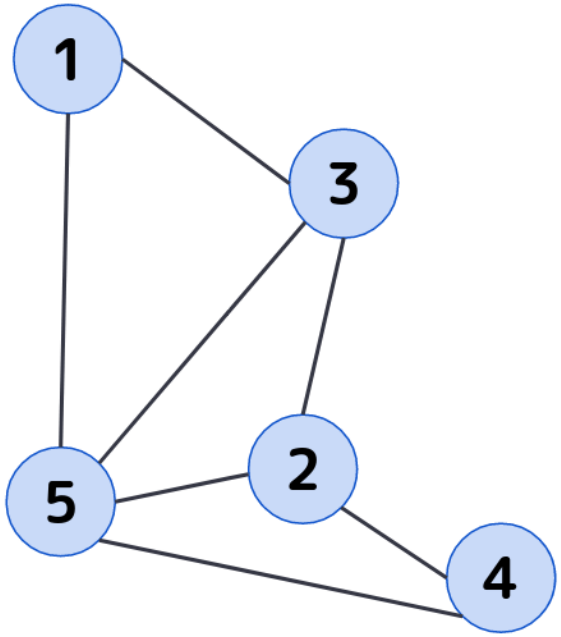
グラフを作る方法

頂点を区別した（頂点に番号がついた）グラフを作ることを考える

- ランダムグラフを作る
- 木を作る



木



グラフ

木を作る方法

- N 頂点の木から、1 から N の整数からなる長さ $N-2$ の数列 (Prüfer sequence) が一意に作れる
- Prüfer sequence から木を作ることができるし、その逆もできる
  - 木を一様ランダムに作るには、1 から N の整数からなる長さ $N-2$ の数列をランダムに作り、木に変換すればよい
  - 木から Prüfer sequence を作る方法は Appendix に記載

以下の説明は 1-indexed であることに注意

Prüfer sequence から木を作る: $O(N \log N)$ Link

```
function prufer_sequence_to_tree(A)
    (d_1, d_2, ..., d_N) ← A での i の出現回数 + 1      // 頂点の次数に対応
    for i = 1, ..., N − 2 :
        d_v = 1 かつ頂点番号が最小の頂点 v を求める
        頂点 v と A_i の間に辺を加える
        d_v と d_i の値を 1 減らす
    end for
    d_u ≠ 0, d_v ≠ 0 となる頂点 u と v の間に辺を加える
end function
```

まとめ

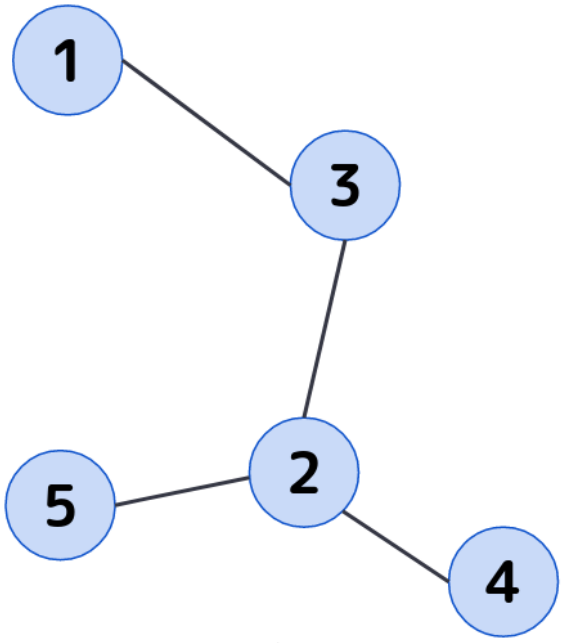
- 今回は、以下を一様ランダムに作る方法を紹介した
- 少しでもバグらせると一様でなくなるので、バグに注意
- より良いランダムケースを作れるようになりましょう！

対象物	計算量
順列	$O(N)$
N 個の中から相異なる K 個を選ぶ	expected $O(K)$ deterministic $O(K \log K)$
ランダムグラフ	expected $O(M)$
木	$O(N \log N)$
(連結ランダムグラフ)	expected $O(N \log N + (M - N))$ ※一様ランダムでない

Appendix: 木から Prüfer sequence を作る

木から Prüfer sequence を作る: $O(N \log N)$

- 2 つの頂点が残るまで、以下を繰り返す
  - その時点での木の葉のうち、番号が最も小さいものを選ぶ
  - 隣接する頂点を Prüfer sequence に追加し、選んだ葉を木から取り除く



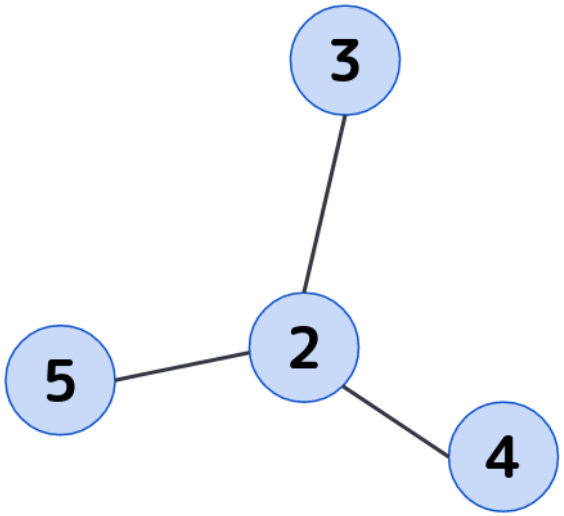
木

列 = [ ]

Appendix: 木から Prüfer sequence を作る

木から Prüfer sequence を作る: $O(N \log N)$

- 2 つの頂点が残るまで、以下を繰り返す
  - その時点での木の葉のうち、番号が最も小さいものを選ぶ
  - 隣接する頂点を Prüfer sequence に追加し、選んだ葉を木から取り除く



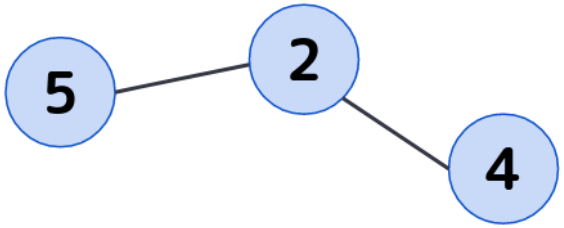
木

列 = [3]

Appendix: 木から Prüfer sequence を作る

木から Prüfer sequence を作る: $O(N \log N)$

- 2 つの頂点が残るまで、以下を繰り返す
  - その時点での木の葉のうち、番号が最も小さいものを選ぶ
  - 隣接する頂点を Prüfer sequence に追加し、選んだ葉を木から取り除く



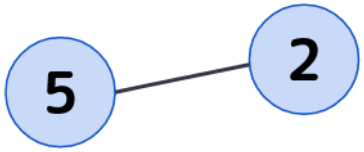
木

列 = [3, 2]

Appendix: 木から Prüfer sequence を作る

木から Prüfer sequence を作る: $O(N \log N)$

- 2 つの頂点が残るまで、以下を繰り返す
  - その時点での木の葉のうち、番号が最も小さいものを選ぶ
  - 隣接する頂点を Prüfer sequence に追加し、選んだ葉を木から取り除く



木

列 = [3, 2, 2]