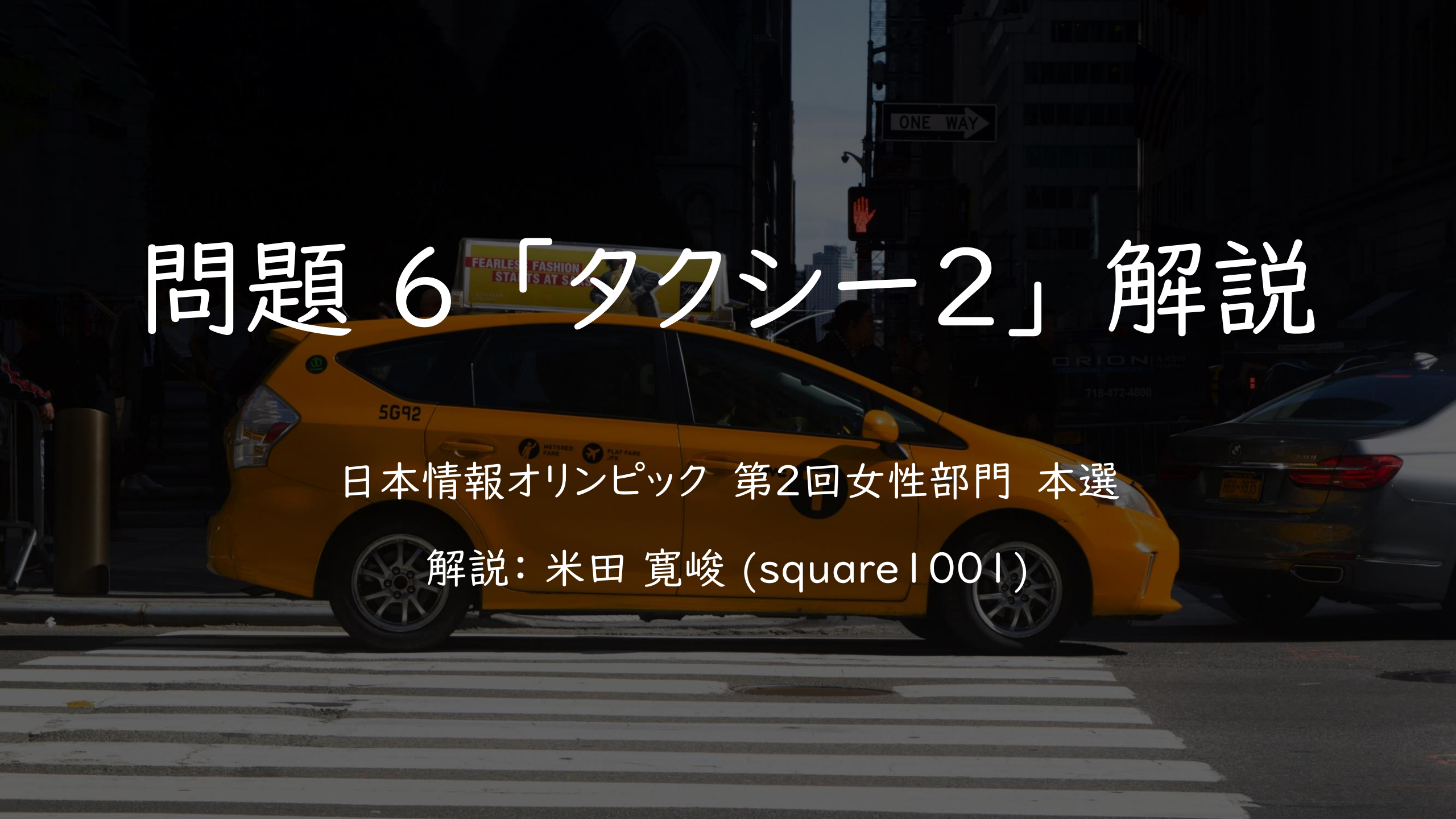



# 問題 6「タクシー2」解説

日本情報オリンピック 第2回女性部門 本選

解説: 米田 寛峻 (square1001)

# 問題の説明



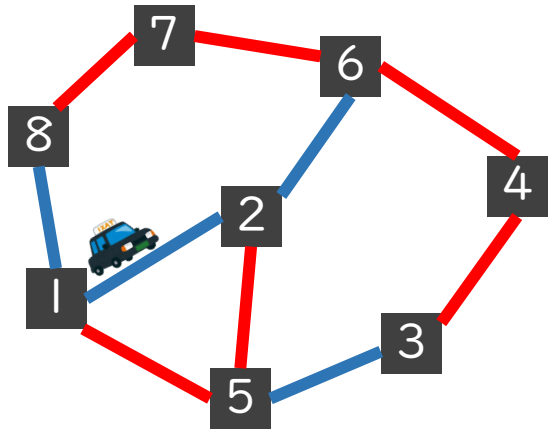
$N$ 個の町と $M$ 本の道（赤色 or 青色）があり、その情報が与えられます

- 赤い道を通ると、所持金が $x$ 円 → $x-1$ 円になります
- 青い道を通ると、所持金が $x$ 円 → $\lfloor x \div 2 \rfloor$ 円になります

「町 $1$ から出発して、町 $T$ まで $1$ 円以上残して到着するためには、最初に何円の所持金を持っておく必要があるか？」

という形式の質問に $Q$ 回答えてください

ただし $L$ 円でも足りない場合は “Large” と答えること



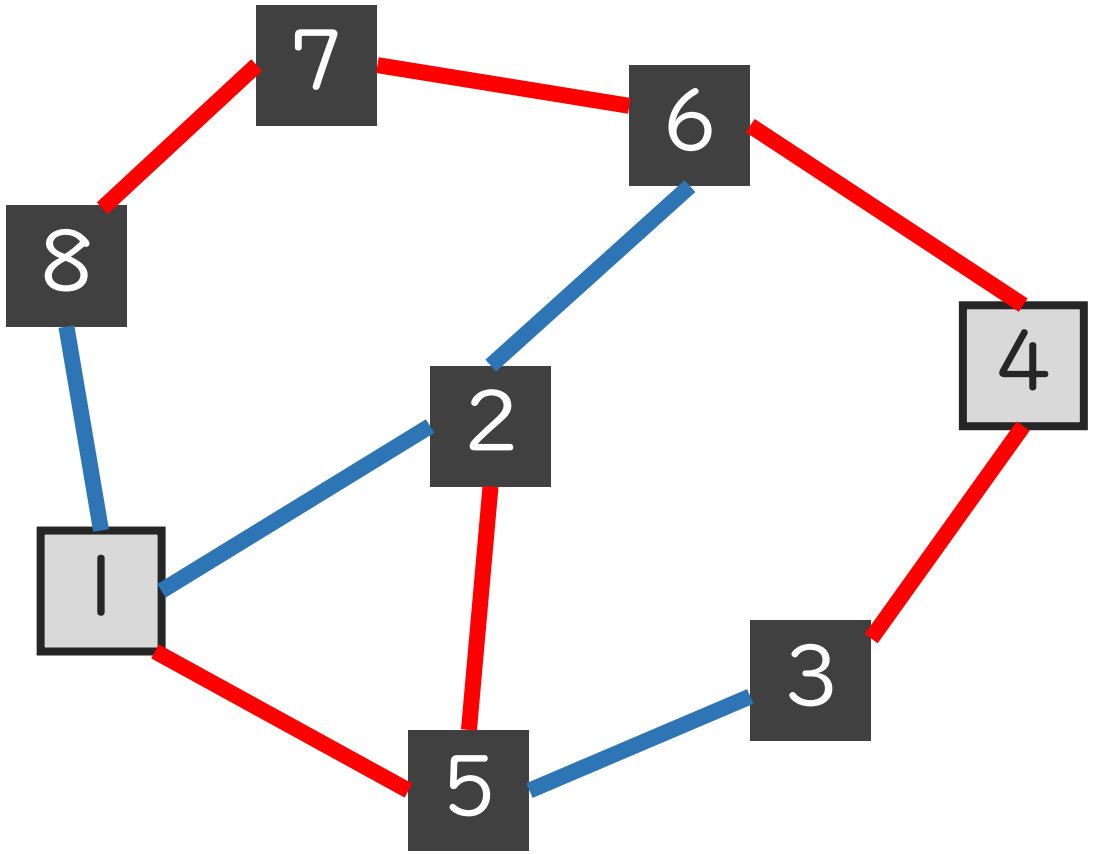
# 具体的な例



右図の例で、町 1 から町 4 に行きたい場合

町 1 → 5 → 3 → 4 の順に行くのが最適

赤色・青色・赤色の道を順に通るので、最初の所持金が 5 円なら OK（お金が 5 → 4 → 2 → 1 円と減っていく）



# 各小課題の制約



この問題には 7 つの小課題があり、できるだけ性能がよいプログラムを書くほど高得点が得られるようになっています（例えば「道路が一直線の場合」を解くと 9 + 19 + 19 = 47 点）

| 小課題（得点） | 道路は一直線か？ | $Q = 1$ か？ | $N, M$ の制約 | その他の制約 |
|---|---|---|---|---|
| #1（9 点） | ✓ | ✓ | 100 | $L \leqq 100$ |
| #2（19 点） | ✓ | ✓ | 200,000 | - |
| #3（19 点） | ✓ | | 200,000 | - |
| #4（16 点） | | ✓ | 50,000 | 道はすべて赤色 |
| #5（20 点） | | ✓ | 50,000 | - |
| #6（12 点） | | | 50,000 | - |
| #7（5 点） | | | 200,000 | - |

# 小課題 1（9 点）

道路網は一直線・$N \leq 100$・$Q = 1$・$L \leq 100$

# 小課題 1 の解法



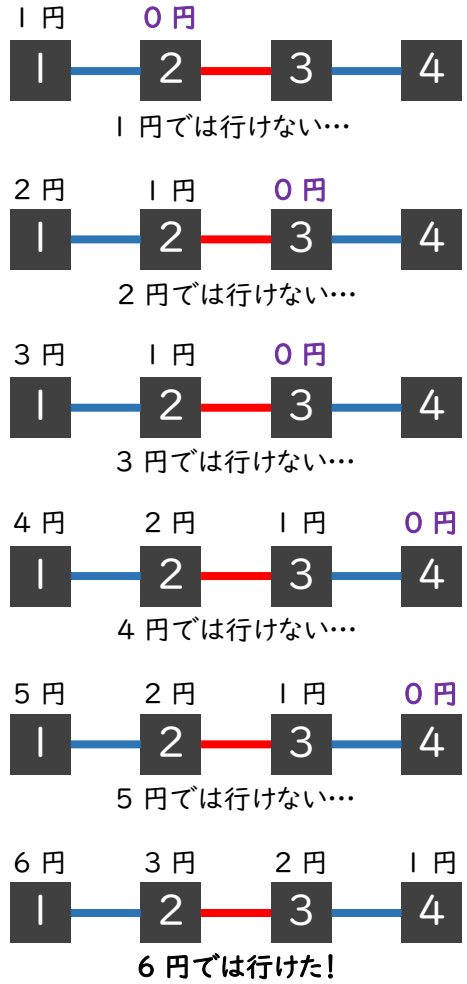
小課題 1～3 では、道が一直線になっています

この場合、タクシーは「町 1 → 2 → … → $T_i$」の順に通ることになるので、所持金が決まれば簡単にシミュレーションできます

以下のアルゴリズムで解くと、小課題 1 に正解できます

## アルゴリズム

- 「1 円でいけるかな？」「2 円でいけるかな？」…「L 円で行けるかな？」と試していったときに、初めて行けたときの所持金が答え
- L 円でも無理だったら、答えは ”Large” になる
- 所持金を決めたときのシミュレーションは計算量 $O(N)$ かかるので全体計算量 $O(NL)$ で解ける




→ 答えは「6」

# 小課題 1 のプログラム例



C++ と Python での実装例です（入力・出力の部分は省略しています）

## C++

```cpp
int answer = -1; // answer = -1 の場合 "Large" とする
for (int money = 1; money <= L; money++) {
    int x = money;
    for (int i = 1; i < T; i++) {
        if (C[i] == 1) x -= 1;
        if (C[i] == 2) x /= 2;
    }
    if (x >= 1) {
        answer = money;
        break;
    }
}
```

## Python

```python
answer = -1 # answer = -1 の場合 "Large" とする
for money in range(1, L + 1):
    x = money
    for i in range(1, T):
        if C[i] == 1:
            x -= 1
        if C[i] == 2:
            x //= 2
    if x >= 1:
        answer = money
        break
```


ここまでで 9 点

# 小課題 2（28 点）

道路網は一直線・$N \leq 200\,000$・$Q = 1$

# 小課題 2 の解法：二分探索



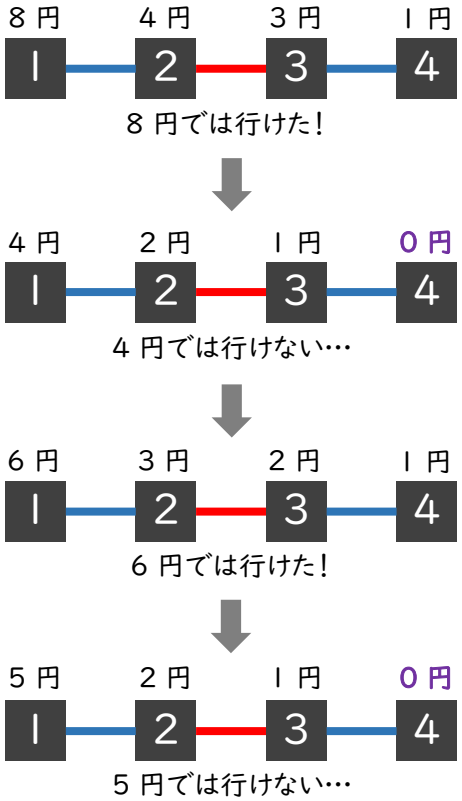
小課題 1 では「1 円で行けるか？」「2 円で行けるか？」… と順々に確かめていったが、このようにするのは非効率的です

ここで、答えを「二分探索」してみましょう

## 答えを「二分探索」とは？

- 例えば、答えが 1～16 であることが分かっていたとする
- 最初の所持金を 8 円としてシミュレーションを行うと…？
  - 行けた場合、答えが 1～8 と分かる
  - 行けなかった場合、答えが 9～16 と分かる
- このようなことを繰り返すと、答えの範囲が 16 個 → 8 個 → 4 個 → 2 個 → 1 個と半分ずつに減っていき、効率的に答えが探せる

具体例：$L = 15$ の場合




→ 答えは「5 円」と分かる

# 小課題 2 の解法：二分探索



二分探索では $\log_2 L$ 回ほどシミュレーションするので、計算量 $O(N \log L)$ で解けました

C++ と Python における二分探索のイメージを載せておきます
※ ここでは、check(x) を「最初の所持金 x 円で行けるか」を返す関数とします

**C++**

```cpp
int l = 1, r = L + 1;
while (r - l >= 1) {
    int m = (l + r) / 2;
    if (check(m) == true) r = m;
    else l = m + 1;
}
if (r == L + 1) {
    cout << "Large" << endl;
}
else {
    cout << r << endl;
}
```

**Python**

```python
l, r = 1, L + 1
while r - l >= 1:
    m = (l + r) // 2
    if check(m) == True:
        r = m
            else:
        l = m + 1
if r == L + 1:
    print("Large")
else:
    print(r)
```


ここまでで 28 点

# 小課題 2 の解法：後ろから考える

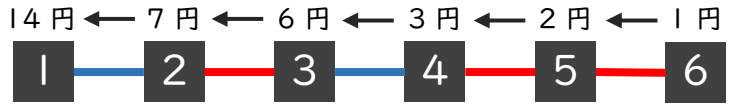


小課題 2 は「後ろから考える」方法で解くこともできます


例えば右図の例で、町 1 から町 6 に行くとき…

- 町 6 で 1 円残すためには、町 5 では最低 1 + 1 = 2 円はないといけない
- 町 5 で 2 円残すためには、町 4 では最低 2 + 1 = 3 円はないといけない
- 町 4 で 3 円残すためには、町 3 では最低 3 × 2 = 6 円はないといけない
- 町 3 で 6 円残すためには、町 2 では最低 6 + 1 = 7 円はないといけない
- 町 2 で 7 円残すためには、町 1 では最低 7 × 2 = 14 円はないといけない

このようにして答えが「14 円」と分かります



# 小課題 2 の解法：後ろから考える



したがって、1 つの質問あたり計算量 $O(Q)$ で解くことができます

C++ と Python での実装例です（入力の部分は省略しています）

**C++**

```cpp
int answer = 1;
for (int i = T - 1; i >= 1; i--) {
    if (C[i] == 1) answer += 1;
    if (C[i] == 2) answer *= 2;
    if (answer > L) break;
}
if (r > L) {
    cout << "Large" << endl;
}
else {
    cout << r << endl;
}
```

**Python**

```python
answer = 1
for i in range(T - 1, 0, -1):
    if C[i] == 1:
        answer += 1
    if C[i] == 2:
        answer *= 2
    if answer > L:
        break
if r == L + 1:
    print("Large")
else:
    print(r)
```


ここまでで 28 点

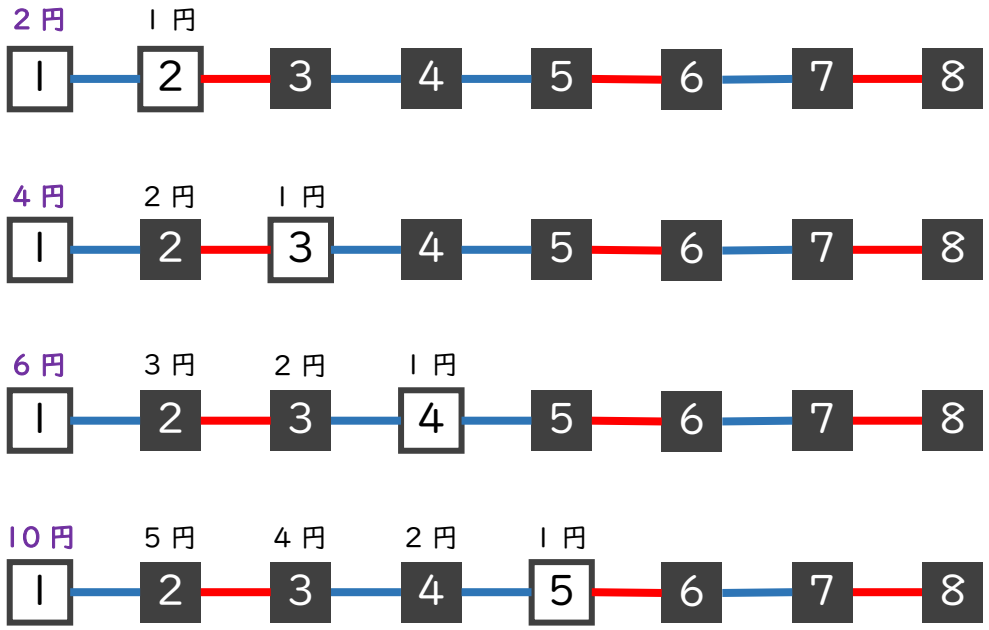
# 小課題 3（47 点）

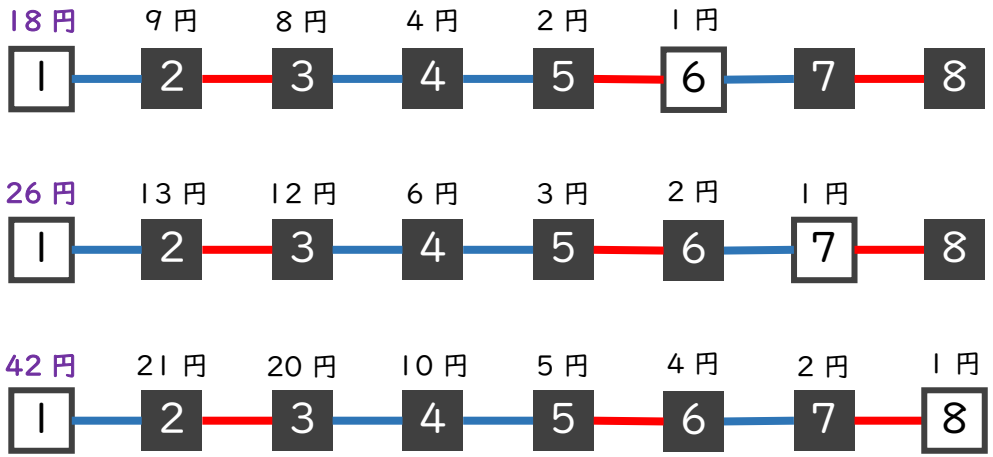
道路網は一直線・$N \leq 200\,000$・$Q \leq 200\,000$

# 小課題 3 の解法



小課題 3 ではたくさんの質問が与えられます
すべての町に対して、答え ＝「必要な所持金の最小値」を求めることになります



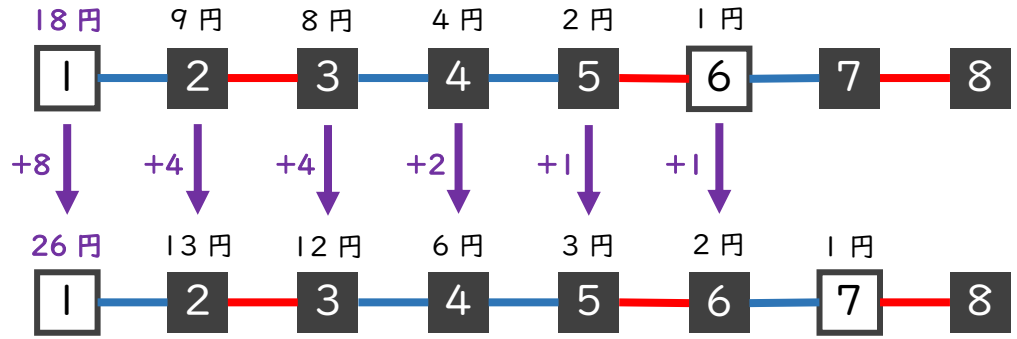



# 小課題 3 の解法



実は、$T = 2, 3, 4, \ldots, N$ のときの答えを順々に計算していくと効率的です

例えば、先ほどの例で $T = 6$ から $T = 7$ に変化したとき、答えがどう変わるか注目しましょう




駅 6 と駅 $i$ までの青い道が $k_i$ 個のとき
駅 $i$ 時点での所持金が $2^{k_i}$ 円上がっている!

このように、$T = j$ から $T = j + 1$ に変化したとき、答えは $2^{(\text{駅 1 と駅 } j \text{ の間の青い道の個数})}$ 円だけ上がります

こうして、すべての $T$ に対する答えが、計算量 $O(N)$ で求められます

# 小課題 3 の解法



C++ と Python での実装例です（入力の部分は省略しています）

## C++

```cpp
ans[1] = 1;
for (int i = 2; i <= N; i++) {
    ans[i] = L + 1;
}
int blues = 0;
for (int i = 1; i < N; i++) {
    // (1 << blues) は 2 の blues 乗
    ans[i + 1] = ans[i] + (1 << blues);
    if (ans[i + 1] > L) break;
    if (C[i] == 2) blues++;
}
for (int i = 1; i <= Q; i++) {
    if (ans[T[i]] > L) {
        cout << "Large" << endl;
    }
    else {
        cout << ans[T[i]] << endl;
    }
}
```

## Python

```python
ans = [ L + 1 ] * (N + 1)
ans[1] = 1
blues = 0
for i in range(1, N):
    ans[i + 1] = ans[i] + (2 ** blues)
    if ans[i + 1] > L:
        break
    if C[i] == 2:
        blues += 1
for i in range(Q):
    if ans[T[i]] > L:
        print("Large")
    else:
        print(ans[T[i]])
```


ここまでで 47 点

# 小課題 4（12 点）

$N, M \leq 50\,000$ ・ $Q = 1$ ・タクシーはすべて赤色

# グラフの問題として考えよう

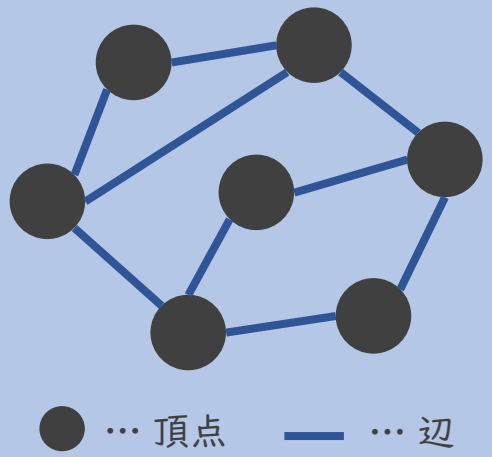


小課題 4 以降は、グラフの問題として解くことになります

## グラフとは？

モノとモノの結びつき方を表すネットワーク構造を **グラフ** といいます

グラフは「頂点」と「辺」からなります
道路網もグラフで表すことができ、町が「頂点」、道が「辺」に対応




詳しく知りたい方はこちら

- 「アルゴリズム×数学」p.162～168
- https://qiita.com/square1001/items/6d414167ca95c97bd8b2 3 節

※ 棒グラフや折れ線グラフなどの「グラフ」とは全く別の概念であることに注意してください

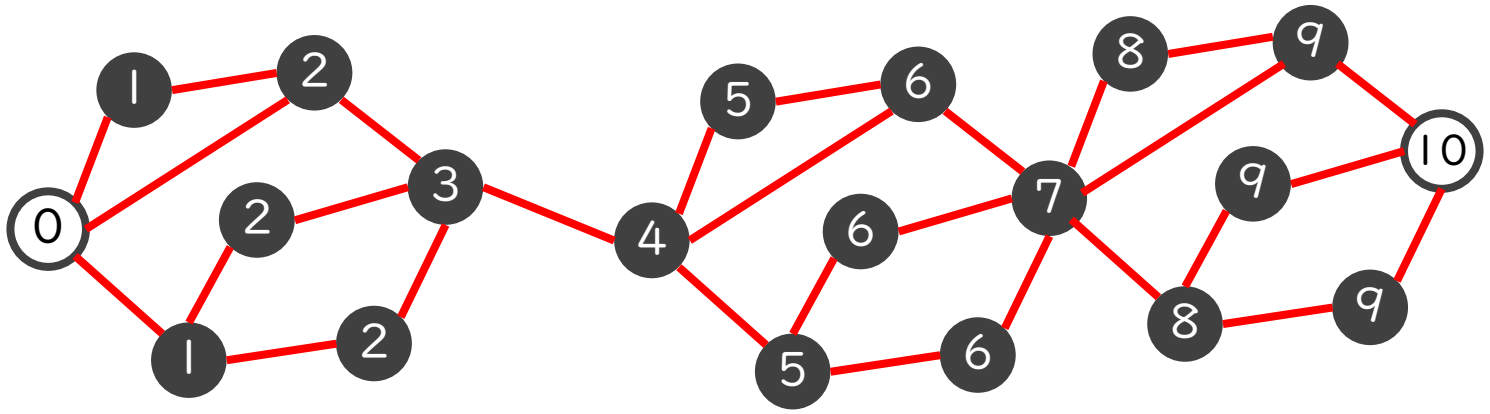
# 小課題 4 の解法



小課題 4 の制約は「$C_i = 1$」すなわちすべての道が赤色です

つまり、頂点 1 から頂点 T まで、最も少ない辺の本数たどって移動する方法を求めたいです

- なぜなら、（必要な所持金）＝（通った辺の本数）＋ 1 だから




（丸の中の数はスタート地点からの最短経路の長さを表す）

# 小課題 4 の解法



実は、(重みなし) グラフの最短経路は「幅優先探索 (BFS)」というアルゴリズムを使うと計算量 $O(N+M)$ で求められます

幅優先探索 (BFS) について知りたい方はこちらを読んでください

- 「アルゴリズム×数学」p.173～177
- https://qiita.com/drken/items/996d80bcae64649a6580

よって、小課題 4 は計算量 $O(N+M)$ で解けました


ここまでで 63 点

# 小課題 5（64 点）

$N, M \leq 50\,000 \cdot Q = 1$

# 小課題 2 を思い出そう



小課題 5 では、赤い辺も青い辺もある場合を解かなければなりません

ここで、小課題 2 の解法を思い出してみましょう

| | |
|---|---|
| 小課題 2: 解法 1 | 答えを二分探索 |
| 小課題 2: 解法 2 | 後ろから考える |

実は、どちらのアイデアも、小課題 5（一般のグラフの場合）の解法につながっています

本解説では、「後ろから考える」アイデアのほうを説明します

※「答えを二分探索」のアイデアを小課題 5 に適用すると計算量が $O(M \log M \log L)$ となるため、実装によっては TLE（実行時間制限オーバー）となる可能性もあります。

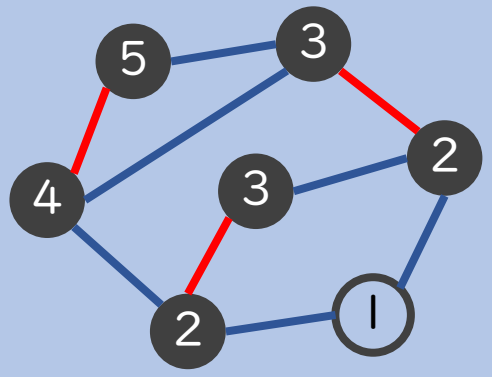
# 小課題 5 の解法



## 後ろから考えた問題

頂点 T からスタートして、頂点 1 でゴールすることを考えます

- 赤い辺を通ると、所持金が +1 されます
- 青い辺を通ると、所持金が ×2 されます




（丸の中の数は、スタート地点から各頂点に到達する時点での最小の所持金）

最初 1 円からスタートするとき、ゴール時点での最小の所持金はいくらでしょうか？

$\mathrm{dist}[v]$ ＝（頂点 $v$ にいる際の最小の所持金）とすると、$\mathrm{dist}[1]$ がこの問題の答えです

これは、どうやって求めるのでしょうか？

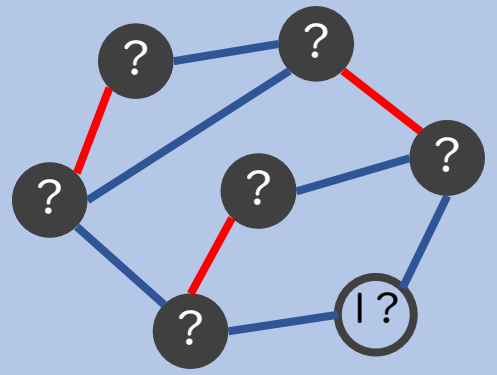
# 小課題 5 の解法



実は、これはダイクストラ法（重み付きグラフの最短経路問題を解くアルゴリズム）と同様のアルゴリズムで求めることができます

## アルゴリズム

各頂点は「距離が未確定の頂点」または「距離が確定した頂点」のいずれかであり、最初はすべて未確定とする



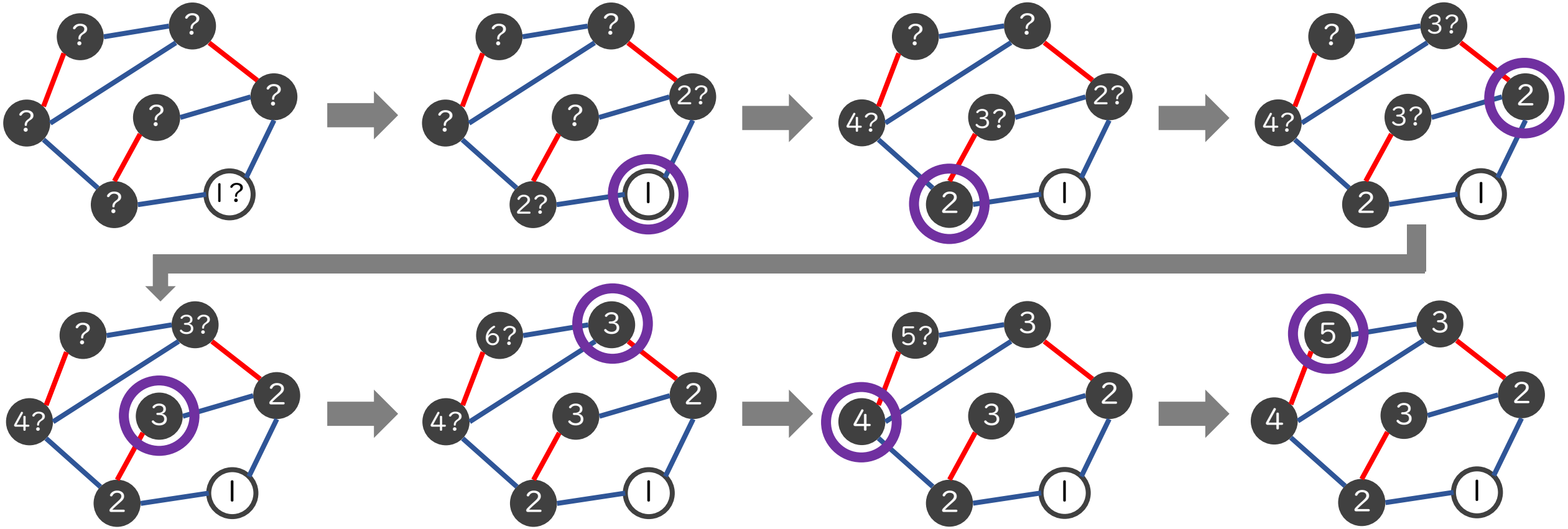
スタート時点での状態

1. 最初 $\mathrm{dist}[T] = 1$ とし、それ以外の頂点の距離の暫定値を $\mathrm{dist}[i] = \infty$ とする（すべて未確定）
2. 以下の処理を $N$ 回繰り返す
   - 未確定の頂点のうち $\mathrm{dist}[v]$ が最小の頂点 $v$ を選び、距離を確定させる
   - 頂点 $v$ と隣接する頂点の $\mathrm{dist}[i]$ よりも、$dist[v] + 1$（赤辺の場合）あるいは $\mathrm{dist}[v] \times 2$（青辺の場合）の方が小さければ、$dist[i]$ を更新する
3. 結果として、$dist[1], dist[2], \ldots, dist[N]$ がすべて求まった（答えは $dist[1]$）

# 小課題 5 の解法



前ページの例では、アルゴリズムは以下のように動作します



# 小課題 5 の解法



未確定の頂点のなかで $\mathrm{dist}[v]$ が最小のものを取り出すのに「優先度付きキュー」を使うと
全体計算量 $O(M \log M)$ でこの問題が解けます

説明が難しかったかもしれませんが、ダイクストラ法とほぼ同じアルゴリズムです

ダイクストラ法について知りたい方はこちら

- 「問題解決力を鍛える！ アルゴリズムとデータ構造」p. 254～264
- https://www.youtube.com/watch?v=e6X2gDTZYCQ（早大・早水桃子先生の動画）


ここまでで 83 点

# 満点解法（100 点）

$N, M, Q \leq 200\,000$

# 小課題 3 を思い出そう

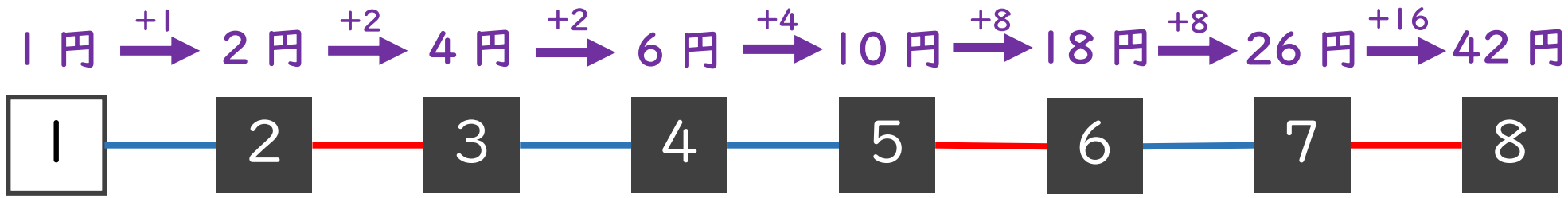


小課題 6・7 ではたくさんの質問が与えられます

$T = 2, 3, \ldots, N$ すべてに対し、答え ＝「必要な所持金の最小値」を求めることになります

小課題 3 の解法を思い出してみましょう

前から順に答えを計算していき、辺を通るごとに必要な所持金が $2^{(\text{通った青い道の個数})}$ 円増えます



# 満点解法 A



つまり「通った青い辺の個数」によって、次の頂点に移動するコストが決まります

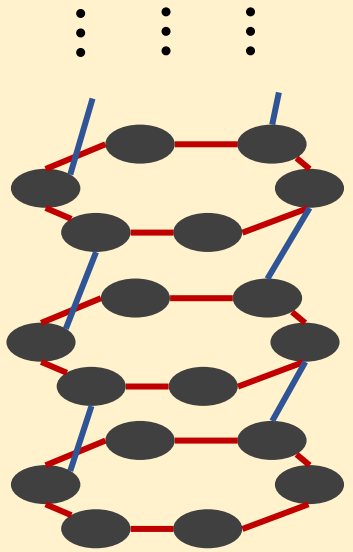
$\mathrm{dist}[level][pos]$ ＝（青い辺を $level$ 個通って頂点 $pos$ に行くときの、最初の所持金の最小値）とすると、以下のように最短経路問題に帰着できます

## 最短経路問題にする

$\mathrm{dist}[level][pos]$ の状態を「状態 $(level, level)$」ということにする

- 頂点 $pos$ と頂点 $i$ を赤い辺がつないでいるとき、状態 $(level, pos)$
  → 状態 $(level, i)$ にコスト $2^{level}$ で行ける
- 頂点 $pos$ と頂点 $i$ を青い辺がつないでいるとき、状態 $(level, pos)$
  → 状態 $(level + 1, i)$ にコスト $2^{level}$ で行ける

最終的な答えは $\mathrm{dist}[0][T], \mathrm{dist}[1][T], \mathrm{dist}[2][T], \ldots$ の最小値



# 満点解法 A



$N$ 頂点 $M$ 辺の重み付きグラフの最短経路は、ダイクストラ法を使って計算量 $O(M \log M)$ で求められます　※ 重み付きグラフとは、辺ごとにコスト（重み）が異なるグラフのことです

ダイクストラ法を行うと、スタートの頂点 $s$ が決まっているときの頂点 $i$ までの最短距離 $dist[i]$ が、$i = 1, 2, \ldots, N$ すべてに対して求まります

この問題では…？

$level = 0, 1, 2, \ldots, K-1$ に対して最短経路を求める場合、$NK$ 頂点 $MK$ 辺程度のグラフの最短経路を求めるから、計算量 $O(MK \log MK)$ かかります

青い辺は最大で $M$ 個あるから、$K = M$ で計算量 $O(M^2 \log M)$ …？

# 満点解法 A



実は「青い辺を 29 回までしか通らない」を仮定してもよいです！

この理由は、もし青い辺を 30 回以上通れば、距離が L ($\leqq 10^9$) を超えてしまうからです
このとき結果が “Large” になるから、計算しても無駄です

$K = [\log_2 L]$（あるいは $K = 30$）にしてよいので、最短経路問題を計算量 $O(M \log M \log L)$ で解くことにより、この問題のすべての $T$ に対する答えが求まります

※ 上手く実装すれば、C++ や Java などの言語で実行時間制限 4 秒に間に合い、満点を得ることができます。多少遅い実装をしても、95 点が得られます。


ここまでで 100 点

# 満点解法 B



この問題は、計算量 $O(M \log L)$ で解くこともできます

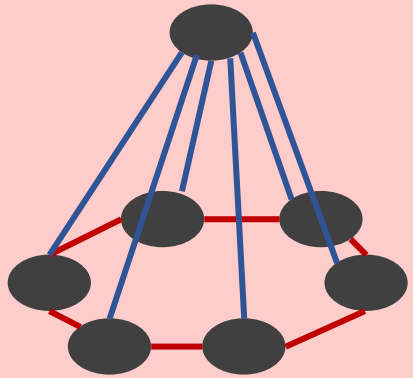
$level$ が同じ $N$ 頂点の「層」だけを考えると、すべての辺のコストは $2^{level}$ で等しいです

したがって、各層ごとに $\mathrm{dist}[level][1], \mathrm{dist}[level][2], \ldots, \mathrm{dist}[level][N]$ を求めると、ダイクストラ法ではなく幅優先探索 (BFS) のように解くことができます

## 解きたい最短経路問題

$N$ 頂点 $M$ 辺で、辺のコストがすべて $C$ のグラフを考える

- このグラフに頂点 $N+1$ を追加し、各 $i$ に対して頂点 $i$ と頂点 $N+1$ をコスト $w_i$ の辺でつないだ新しいグラフを作る
- このグラフに対して、頂点 $N+1$ から各頂点までの最短距離を求めたい



# 満点解法 B



実は、ダイクストラ法を 2 つのキュー $Q_1, Q_2$ を使って再現できます

- $Q_1$：頂点 $N+1$ からコスト $w_i$ の辺を通ったときに push されるキュー
- $Q_2$：頂点 $N+1$ 以外から辺を通ったときに push されるキュー

$Q_1$ と $Q_2$ が最短距離順に並べられているならば、各ステップでは $Q_1$ の先頭と $Q_2$ の先頭のうち $\mathrm{dist}[i]$ が小さい方を取り出せばよいから、$w_1, w_2, \ldots, w_N$ のソート順が分かっている状態であれば計算量 $O(N+M)$ で前述の最短経路問題が解けます

したがって、以下の手順で、全体計算量 $O(M \log L)$ で解けます

1. まず $level = 0$ の層の最短経路を BFS で求める。$dist[0][1], dist[0][2], \ldots, dist[0][N]$ のソート順は、キューが pop された順番で分かる。
2. 次に $level = 1$ の層の最短経路を前述の方法で求める。$dist[1][1], dist[1][2], \ldots, dist[1][N]$ のソート順は、キューが pop された順番で分かる。
3. （中略）
4. 最後に $level = 29$ の層の最短経路を前述の方法で求める。

この方法を使うと、特に C++ では実行時間制限の 10 分の 1 未満しか使わずに正解することもできます