JOI 2015/2016 春合宿 Day4 雪降る道路(Snowy Roads) 解説

2016/3/23

解説担当 : 城下慎也 (IOI 2011 タイ大会 銀メダル)

最初に

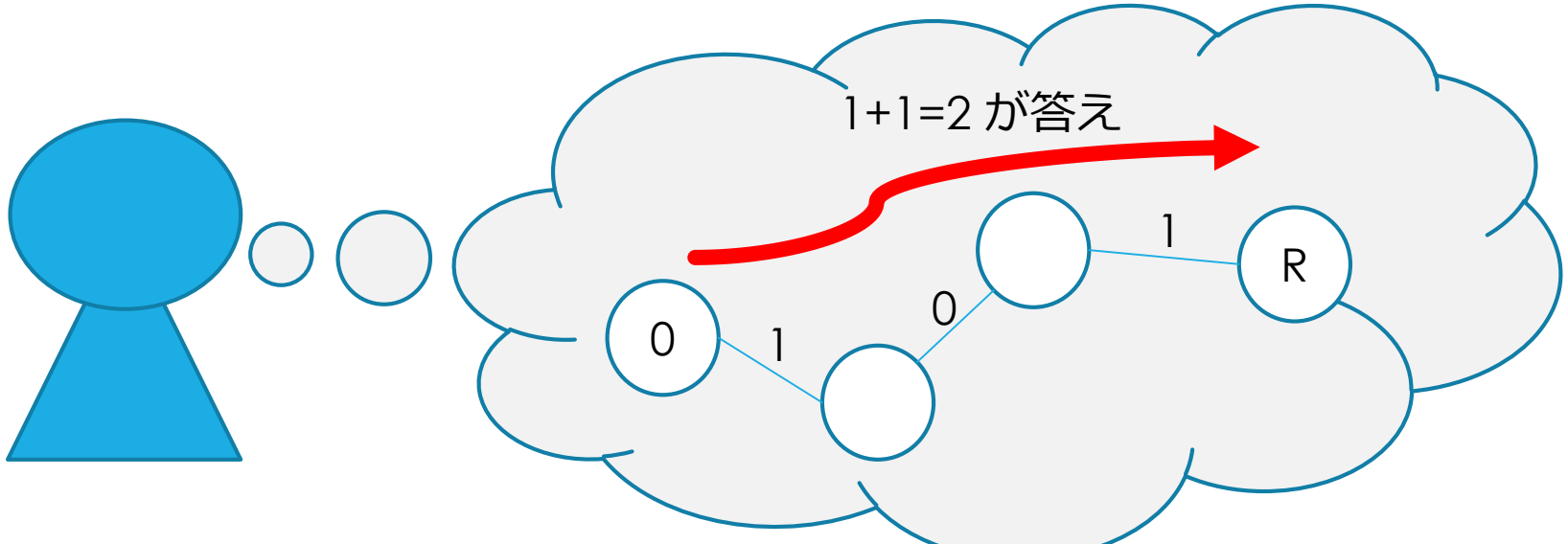
- **問題文はとても長いです。**
- 形式が特殊なので理解等が大変だったかもしれませんが、簡単な部分点等も含まれているので、今回のセットでは時間をかけてでもしっかり内容を理解をしましょう。

問題概要

- Anya と Boris がいて、Anya が中継サーバに保存し、Boris がそこから読みだして質問に回答していきます。
- 中継サーバには容量制限、質問回数制限があります。
- 質問は、都市 0 と別のある都市まで移動するのに経由する、雪降る道路の本数の最小値がいくつか、というものです。
- 同じ日に質問が複数あることもあります。
  - ちなみに、同じ日の 2 回目以降の質問は以前のサーバから情報を活用できそうですが、直前に変更があるかもしれないので、実質的には活用できません。

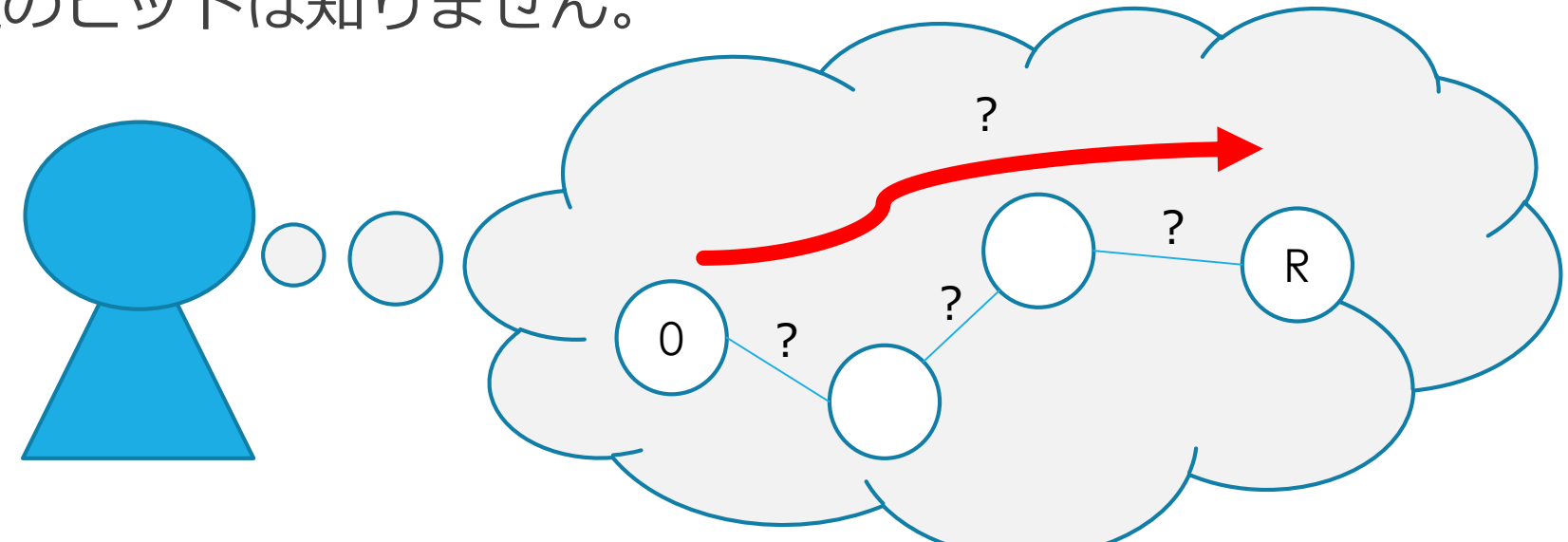
どういうこと？

- Boris は、都市 0 からある都市 R まで到達するために経由する必要のある雪が降っている道路の最小値を答えます。
- 木なのでパスは一意に定まります。

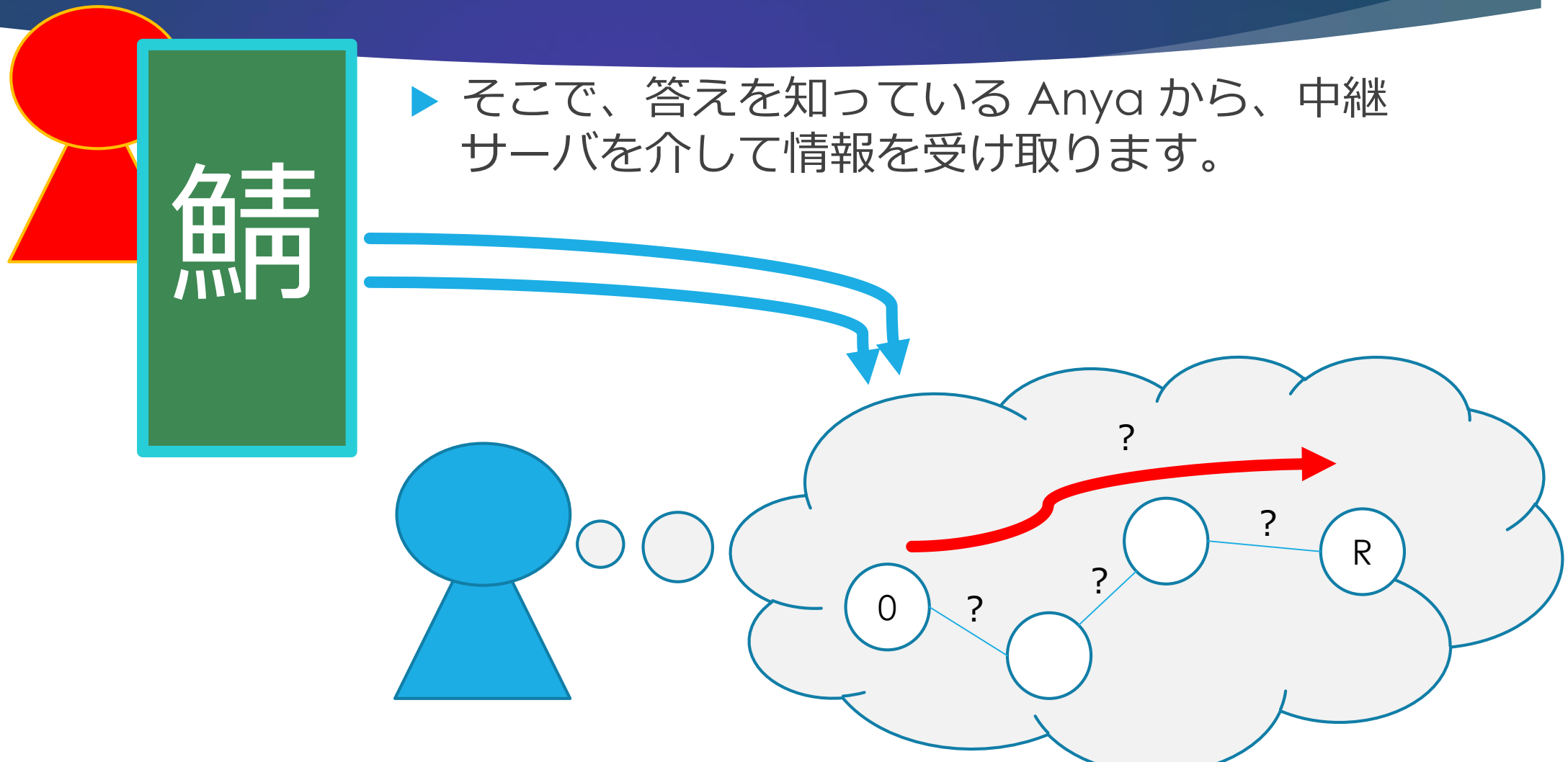


どういうこと？

- Boris は、都市 0 からある都市 R まで到達するために経由する必要のある雪が降っている道路の最小値を答えます。
- 木なのでパスは一意に定まります。
- 困ったことに各辺のビットは知りません。



どういうこと？



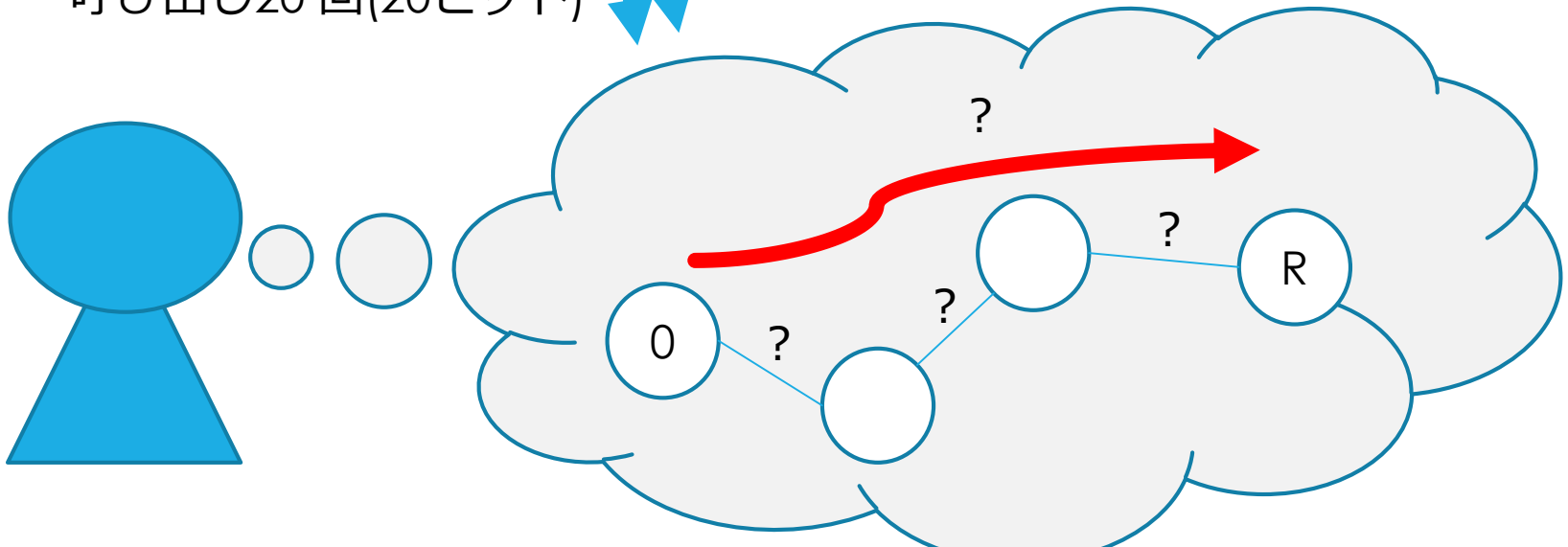
- そこで、答えを知っている Anya から、中継サーバを介して情報を受け取ります。

どういうこと？

容量1,000 ビット

- そこで、答えを知っている Anya から、中継サーバを介して情報を受け取ります。
- ただし、サーバや Ask に制限があります。

呼び出し20 回(20ビット)



小課題 1 (15 点)

- $N \leqq 20$

小課題 1 の特徴

- N がすごく小さい。
- TLE用？ でもパスをたどるのはあまり時間はかからないような…？

- 多方面から考察してみましょう。

小課題 1 以前に

- そもそも、サーバに直接答えを置いたらダメなのでしょうか？
- 降雪情報は全体でも 499 ビットなので、 1,000 ビット制限に引っかかりません。
- Anya が降雪情報をそのまま保存し、Boris が普通にパスを辿って計算すれば良いのでは？

→小課題 1 のみ通ります。

どうして？

- パスが 20 より長いと、Ask 呼び出し制限に引っかかって不正解になります。
- 小課題 1 はパスが短いので、パスが 19 を超えることがなく、この手法でうまくいきます。

小課題 2 (5+15 点)

- $N \leqq 100$

小課題 2 の特徴

- N がそこそこ。
- さすがに小課題 1 用の方針では厳しいです。

小課題 2 以前に

- そもそも、Boris が直接降雪情報を知る必要はあるのでしょうか？
- Boris が答える値は 0 以上 $N-1$ 以下で、全体でも $500(<2^9)$ 通りの答えしか取りません。
- すべて知っている Anya が予め答えを保存し、Boris が答えを読み込んでそのまま送れば良いのでは？

→小課題 1 と 2 が通ります。

どうして？

- サーバの容量制限に引っかかります。
- 各頂点につき 9 ビット消費するので、$9N$ ビット全体でかかることになり、小課題 3 以降では 1,000 ビット制限に引っかかります。

- 小課題 2 なら、 100 頂点以下なので、覚えるビット数は 900 ビットで済みます。

小課題 3 (35 点)

- 直線

2 つの解法を比較してみる

- 先ほどの解法には、それぞれ長所短所があります。

- Anya サボる
- Boris 頑張る
- 記憶量少ない
- アクセス多すぎ

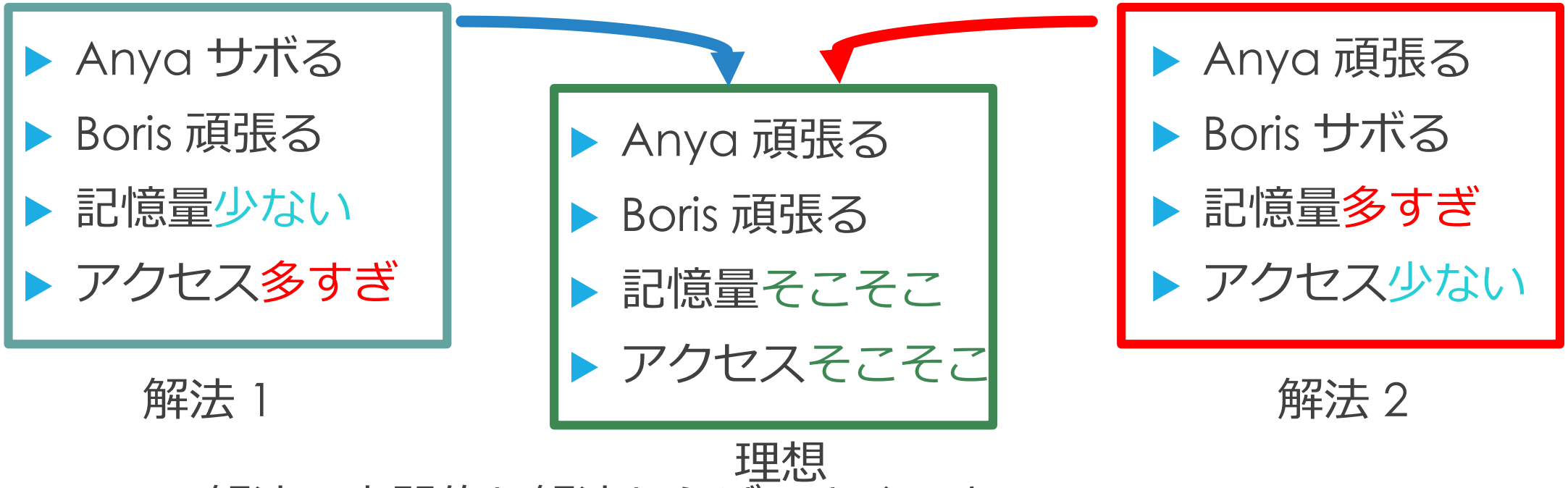
解法 1

- Anya 頑張る
- Boris サボる
- 記憶量多すぎ
- アクセス少ない

解法 2

2 つの解法を比較してみる

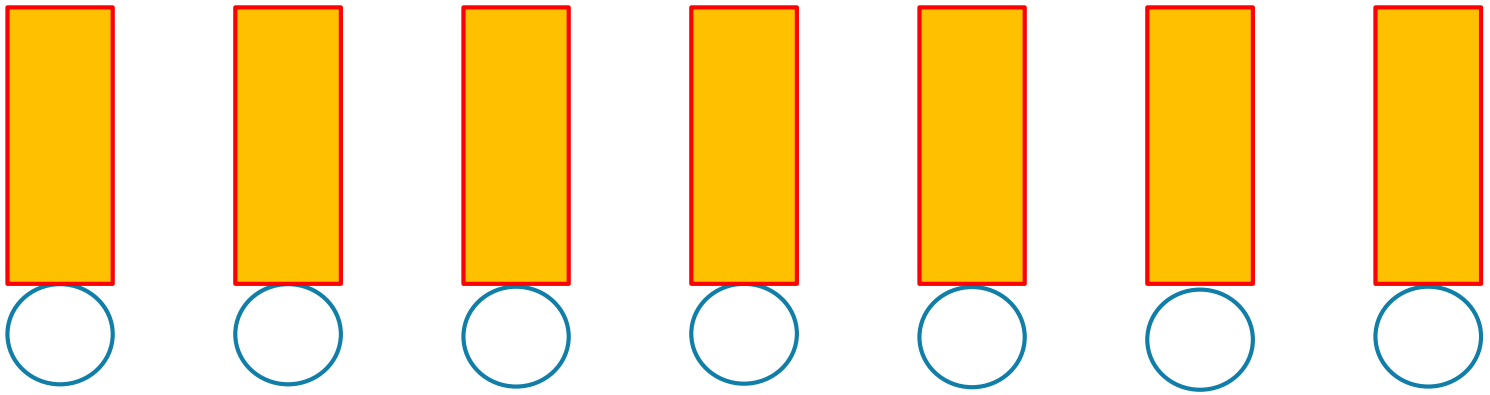
- 先ほどの解法には、それぞれ長所短所があります。



- 2 つの解法の中間的な解法ならばうまくいきそう？

解法２の観察

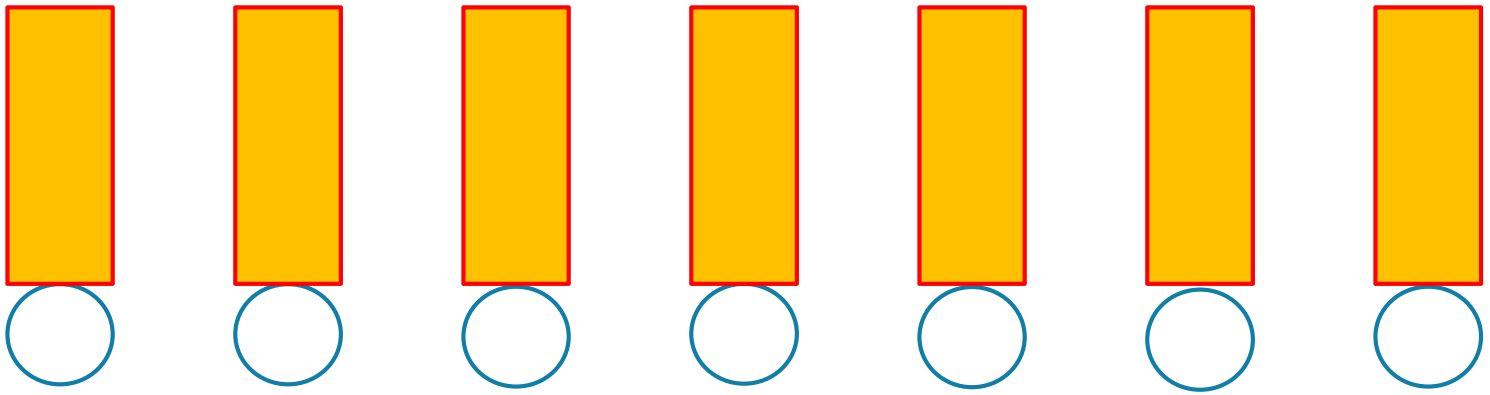
- 全頂点にビットを保存するのは、多くのメモリを消費します。



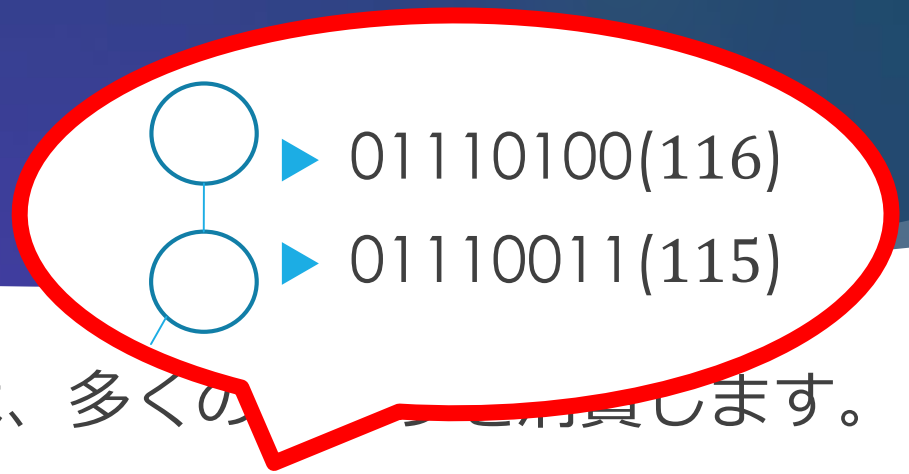
解法２の観察

- 01110100(116)
- 01110011(115)

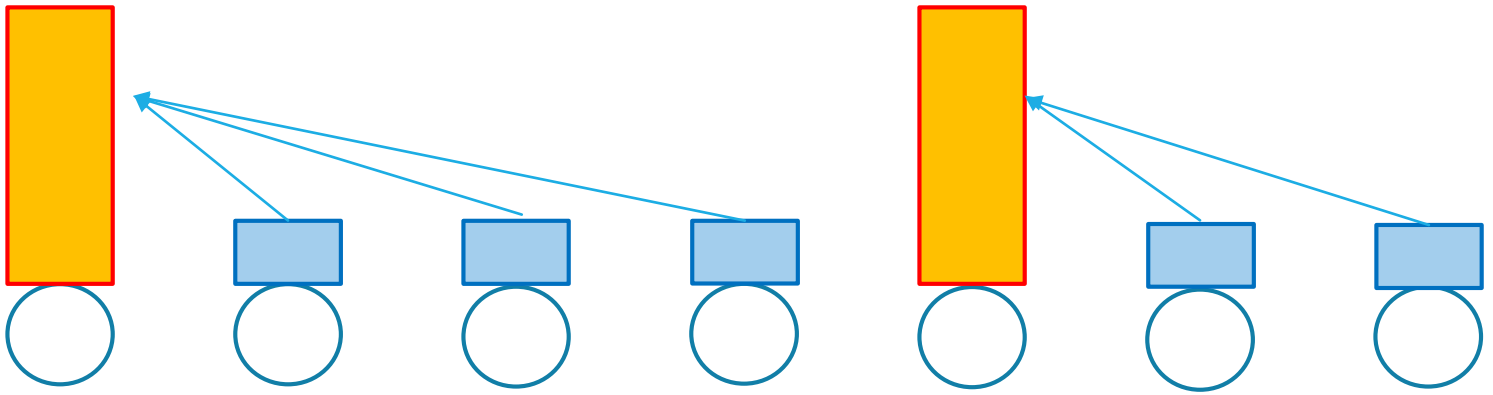
- 全頂点にビットを保存するのは、多くの[illegible]を消費します。
- そもそも、隣り合う頂点の解は高々１しか変わりません。



解法２の観察

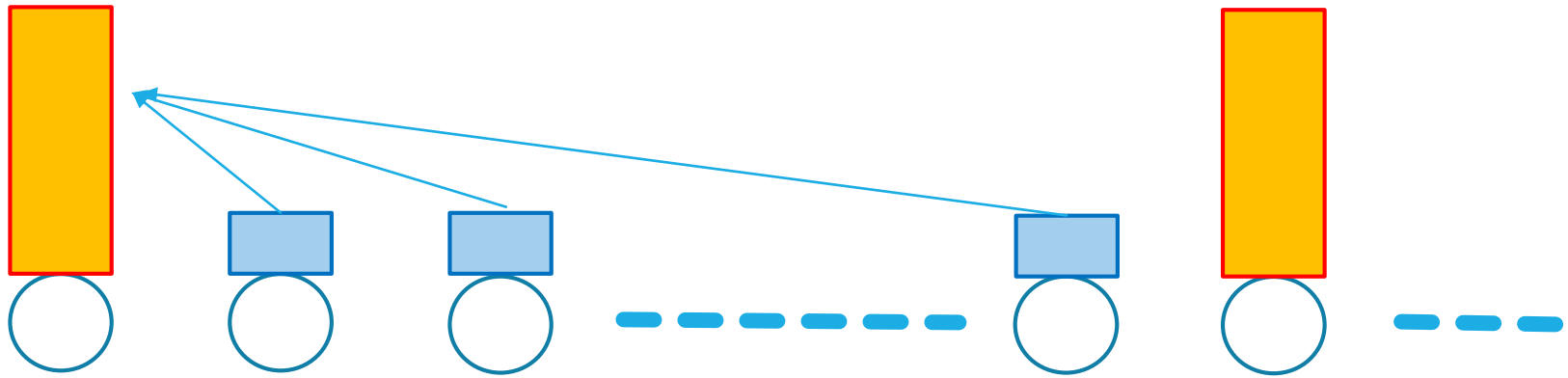


- 全頂点にビットを保存するのは、多くの[illegible]を消費します。
- そもそも、隣り合う頂点の解は高々１しか変わりません。
- そこで、近い位置関係にある頂点に、差分を保管することで節約することにします。



具体的な方針

- 例えば、12 個周期で覚える頂点を定めます。((500/12)×9≒410ビット)
- 残りの頂点には、近くの覚える頂点との差分(実際には辺の情報を覚えておいて質問に対して実際にたどれば良いです。)を覚えます。 (約499ビット)
- 合計で 909 ビットとなり、小課題 3 を解くことができます。

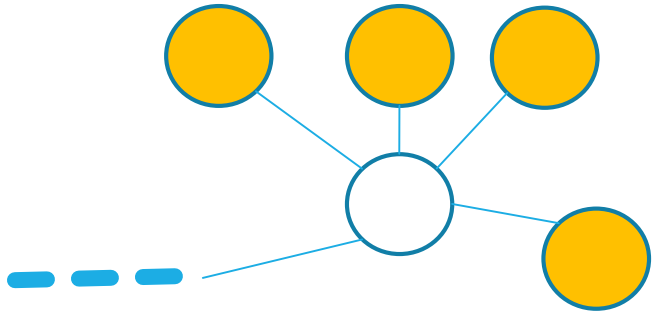


小課題 4 (15+5+35+45 点)

- 追加の制限なし

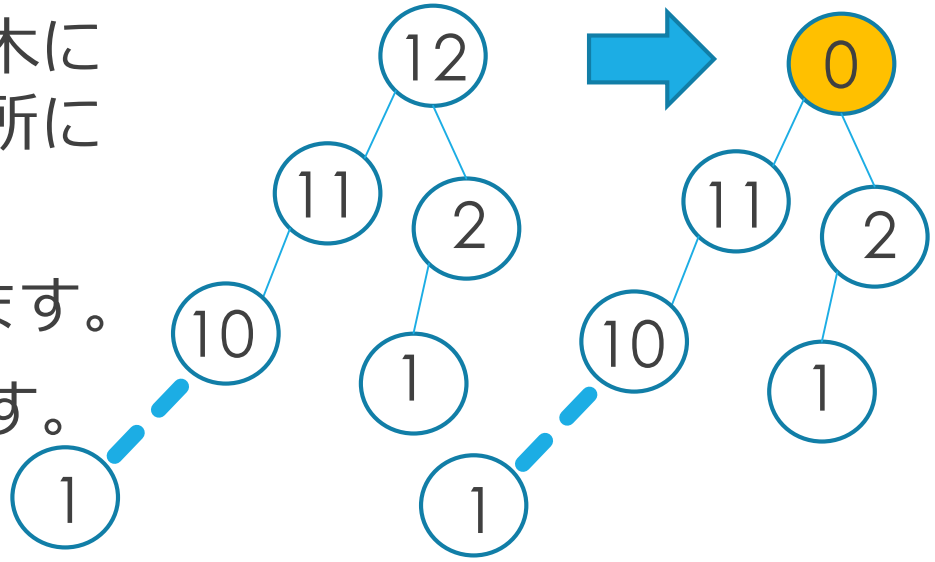
目的

- 先ほどの手法を木上で実装したい。
- ただ、適当に根からの距離で設定すると、同一距離が大量にあるケース (e.g. 星グラフ) で大量に保存してしまいます。
- うまく保存する場所を決めて保存する箇所を節約したいです。
- 以下、都市 0 を根ノードとして探索することにします。



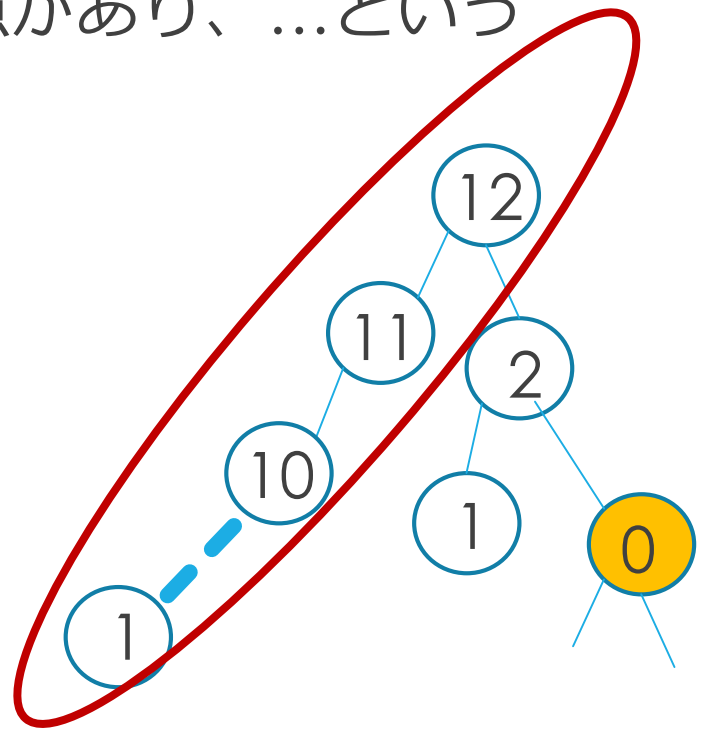
アルゴリズムの例

- 深さ優先探索の戻りがけのタイミングで、含めないと答えられなくなる頂点が子孫に登場するなら、その頂点の解を覚えるようにします。
- 例えば、各ノードに、ここより下の部分木にあるノードのうち、未解決で最も遠い場所にあるノードとの距離を計算します。
- その距離が 12 になれば覚える側に含めます。
- こうすることで満点を得ることができます。



証明

- 12 になって覚える必要のある頂点が登場したとき、その子供には 11 の頂点があり、さらにその子供には10の頂点があり、…というようにして、長さ 12 のパスが取れます。
- そのため、覚える頂点に数えるようにした際、必ず 11 個以上の頂点に関して覚えなくて済むことが保証できます。
- 唯一の例外は都市 0 ですが、都市 0 の値は常に 0 なので気にしなくて大丈夫です。
- 42 (≒500/12) 個程の頂点記憶で足ります。

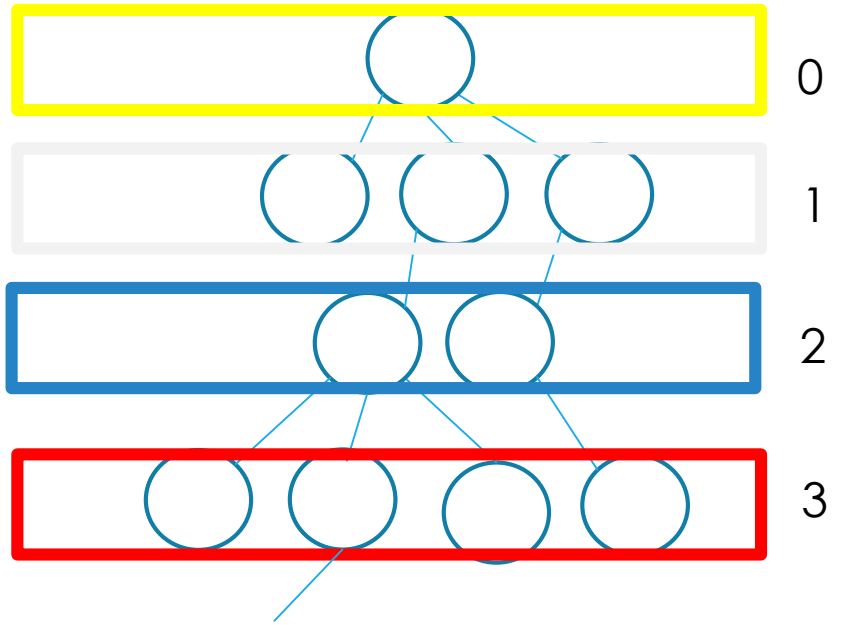


別解

- 深さが mod 12 で一定の値となった場所を覚えることにします。
- もちろん、先ほどの星グラフに衝突した場合失敗します。
- ところで、深さの mod が k となる頂点の個数を s(k) としたとき、

  $$s(0)+s(1)+\ldots+s(11)\leqq 500$$

  がいえます。
- よって、それらの min に合わせれば 42 個ほどの頂点記憶にできます。



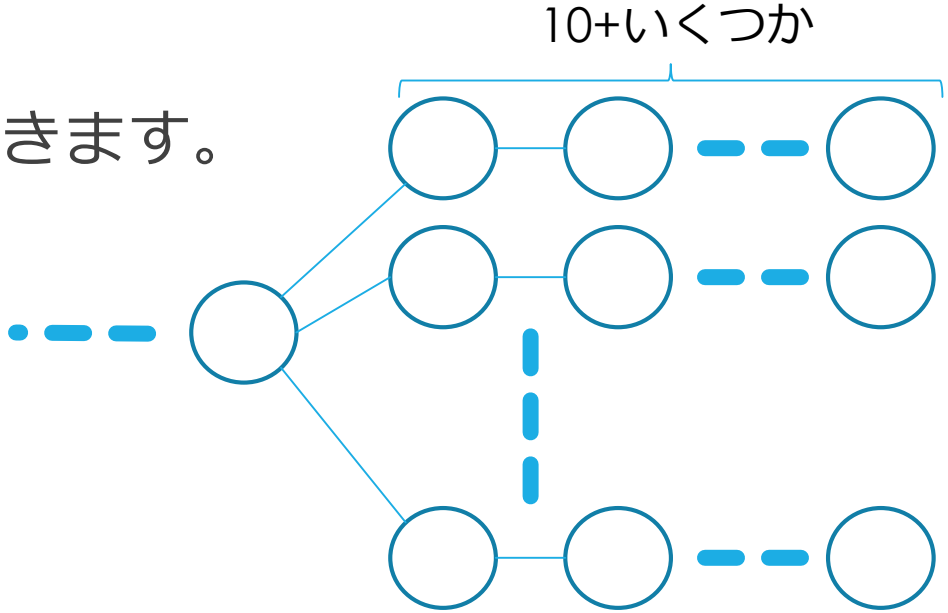
補足

どのように覚える頂点を決定するか

- うまく頂点を設定したいけれども、解析が大変/思いつかない…
- よく考えてみると、子孫側に移動することでも解決することができる！
- そう考えると、42 個くらいでのカバー自体はいっぱいありそう？

乱択アルゴリズム

- 覚える頂点集合を無作為に決定すればうまくいくこともあります。
- しかしながら、適当に決定すると下図のような部分を含んだ際に、失敗率が高くなります。
- 各種最適化で成功率を上げることができます。
- 重要なこととして、ここで紹介したアルゴリズムは**正しいことを保証していません。**
- この辺りを頑張れば、通るかもしれないという程度に思ってください。



乱択アルゴリズム(各種最適化の一例)

- 覚えるビットは、0 以上(都市 0 からの距離) 以下なので、ある程度節約することができます。
- すでにいずれかの到達できる頂点を、次の抽選から除外すると、うまくいきやすくなります。
- 頂点集合を決めれば、今の頂点集合でいけるかは分かるので、もしあるシード値で失敗したときにうまくいくまで別のシード値で試し続ける解法も考えられます。

まとめ

- 長くても問題文をしっかり読み、「何が本質的に問われているのか」を意識しましょう。
- 解法が分からなくても、乱択などを実装してみるとどうにかなることがあるかもしれません。

- もしも他に手を付けられそうになかったならば、試してみる価値はあるかもしれません。

得点分布