人工知能入門
第11回

藤田 悟
黄　潤和

前回の復習

- 幅優先探索と深さ優先探索の比較
  - 2つの探索を書き換える
  - 記憶が必要なオープンリスト数
- 繰り返し深化(Iterative Deepening)
- 分枝限定法

今回学ぶこと

- ヒューリスティック探索
  - 山登り法
  - 最良優先探索
  - 焼きなまし法(Simulated Annealing)
  - A*アルゴリズム

山登り法

山登り法とは「現在の解の近傍の内で最も成績の良い解」を近傍解として選び、「現在の解より近傍解の成績の方が良い場合」に近傍解と現在の解を入れ換える局所探索法の方法。

Hill Climbing: 高い方向に進む

- 山に登るならば、常に高い方向に向かって道を進めば、いずれは頂上に到達する。

しかし、課題も多い

- 自分の登りたい頂上にたどり着くまでには、様々な課題も存在する。

山登り法の派生形

- 単純山登り法 (Hill Climbing)
  - 単峰性があり、道が自由に選べる場合
    - 例:2次関数の最適解探索
- バックトラック付き山登り法
  - 単峰性はあるが、行き止まりがある場合
    - 行き止まりに来たら、バックトラックする
- 最良優先探索 (Best first search)
  - オープンリストの中の最良解(標高が高い、パスが短いなど)を優先して探索する
- 焼きなまし法(Simulated Annealing)
  - 多峰性のある場合
    - 時々、山を下って、周囲を見渡す

https://ja.wikipedia.org/wiki/

最良優先探索（best-first search）は、
幅優先探索（英: breadth-first search）を何らかの規則（評価関数）に従って次に探索する最も望ましいノードを選択するように拡張した探索アルゴリズムである。

探索ノードを効率的に選択するには優先度つきキュー（priority queue）を用いて実装するのが一般的である。キューに貯めずに最良のノードだけを扱うと山登り法になる。キューを評価関数でソートしないと幅優先探索になる。

最良優先探索の例としてはダイクストラ法（Dijkstra's algorithm）やA*アルゴリズム（A* search algorithm）や均一コスト探索を挙げることができる。

最良優先探索は経路探索においてしばしば使われるアルゴリズムである。コンピュータ将棋・コンピュータチェスなどでも最良優先探索を拡張した物が使われている

https://ja.wikipedia.org/wiki/

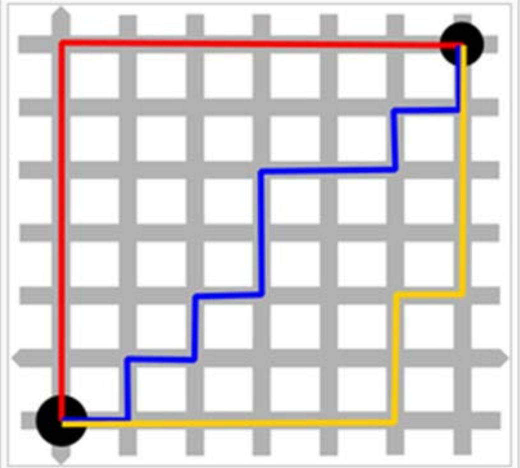
```
function 最良優先探索(startNode)
    visited = 訪問済みノードを管理する集合
    queue = ノード評価関数(EVAL)で自動的にソートするノードの優先度つきキュー

    startNode を visited と queue に追加
    while queue が空ではない do
        node = queue の先頭から取り出す
        if IS_GOAL(node) then
            return node
        for each child in EXPAND(node) do
            if child が visited に含まれない then
                child を visited と queue に追加
    return 探索失敗
```

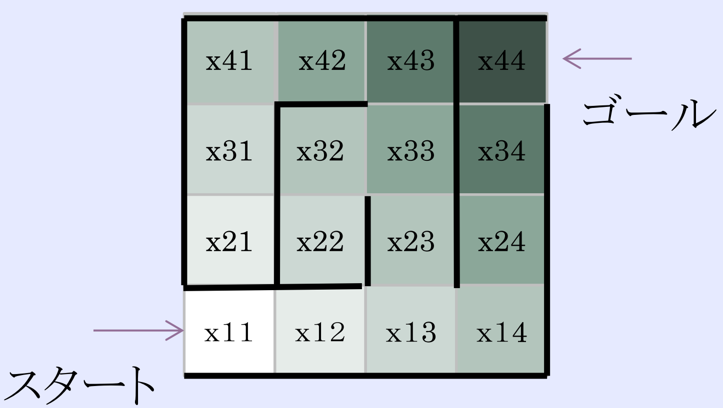
A* の論文では、上記の visited を CLOSED、queue を OPEN と呼ぶ。

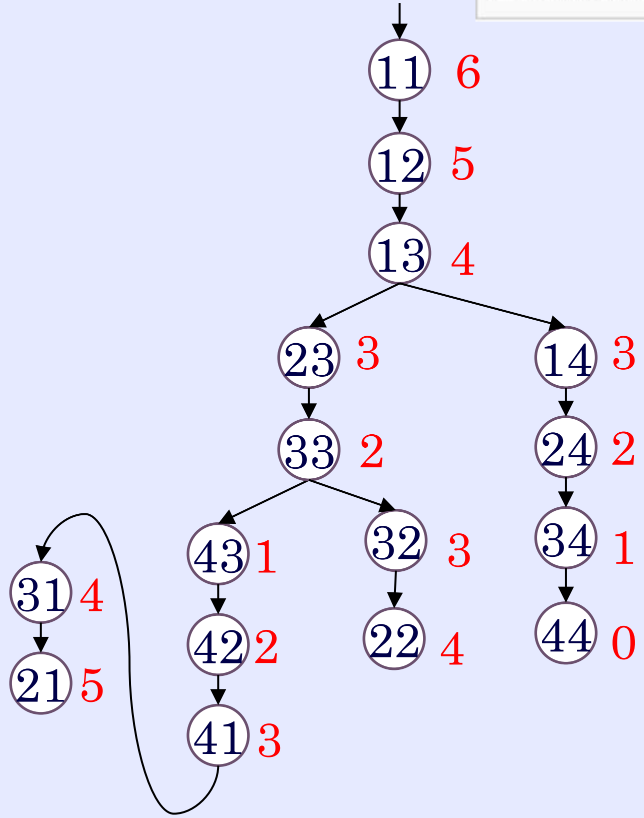
例: 迷路

- ゴールを山の頂上とみなして、ゴールへのマンハッタン距離を標高と考える
  - 行き止まり
  - 一旦、標高を下げる



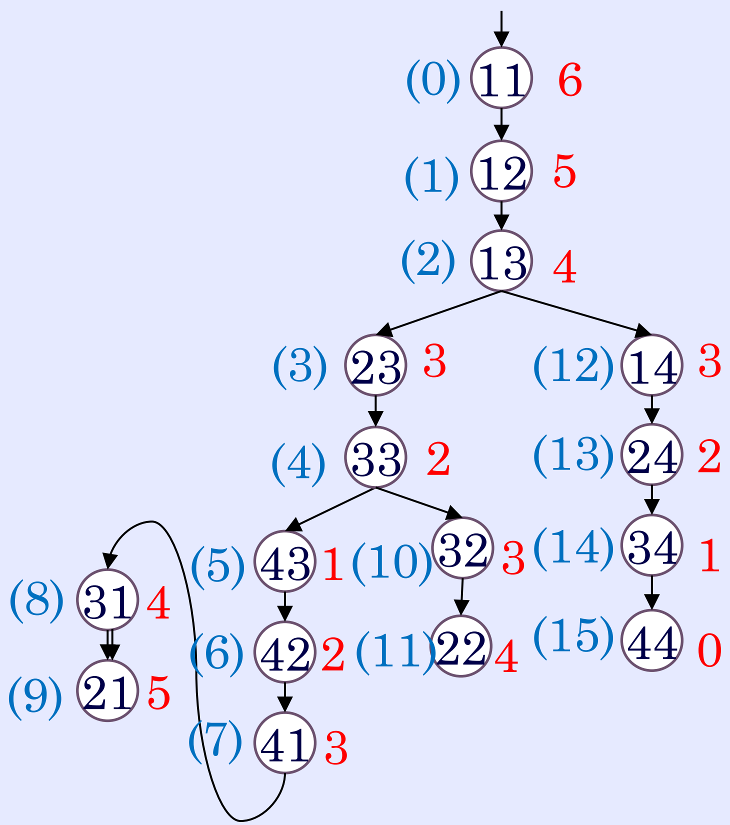
マンハッタン距離の例: どの色のコースを辿っても同じ距離が決まっている



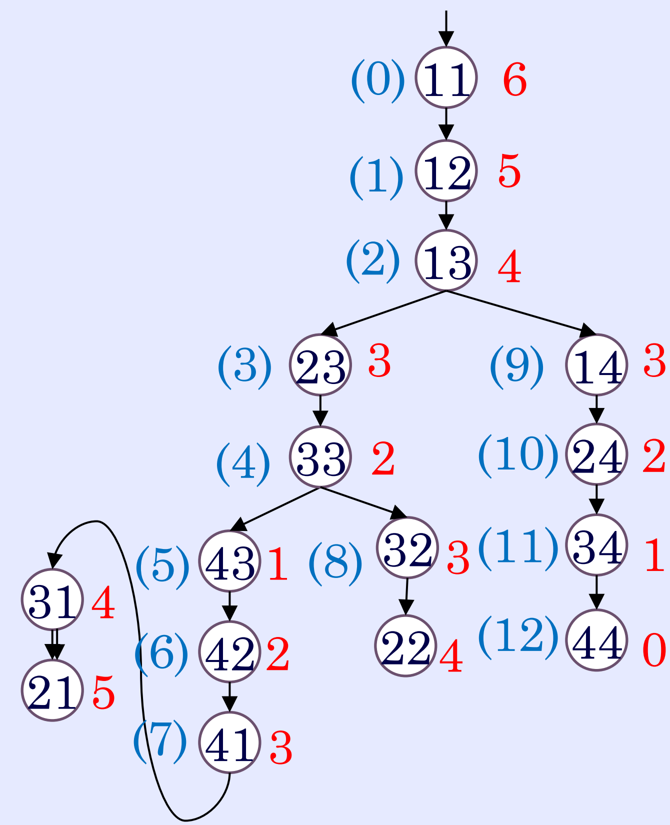


例: 迷路

- バックトラック付き山登り法と最良優先探索では、検索パスが異なる



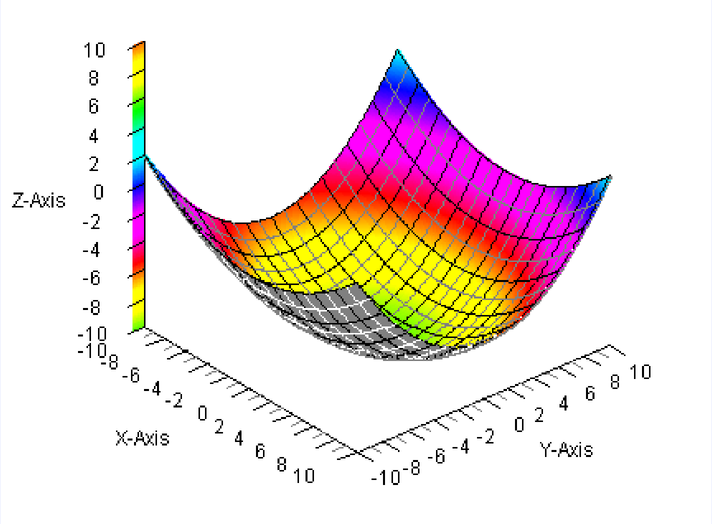
バックトラック付き山登り法



最良優先探索

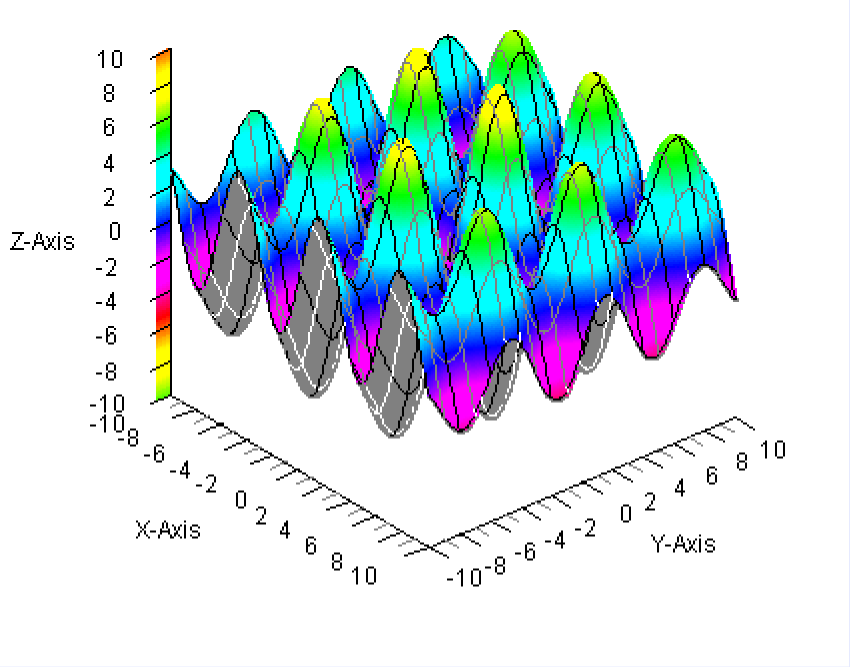
例: 2次関数の最適解探索

- x, y の値を変化させて、結果が小さくなる方向に探索する
  - 探索幅を徐々に小さくしていくとよい
  - 例: 下記の関数を最小とする (x, y)の組み合わせを探索せよ
    - $f(x,y) = \ 0.04\,x^2 + \ 0.08y^2 - \ 10$
    - 山登り法を使って(x, y)の値を変化させ、値が小さくなる方に移動していけば、徐々に (0,0) に近づいていく



多峰性を持つ関数

- 山登り法だけでは最小値/最大値の探索が難しい
  - $f(x, y) = \frac{(4\sin(x)+5\cos(y))}{\left(1+\frac{x}{10}^2+\frac{y}{10}^2\right)}$



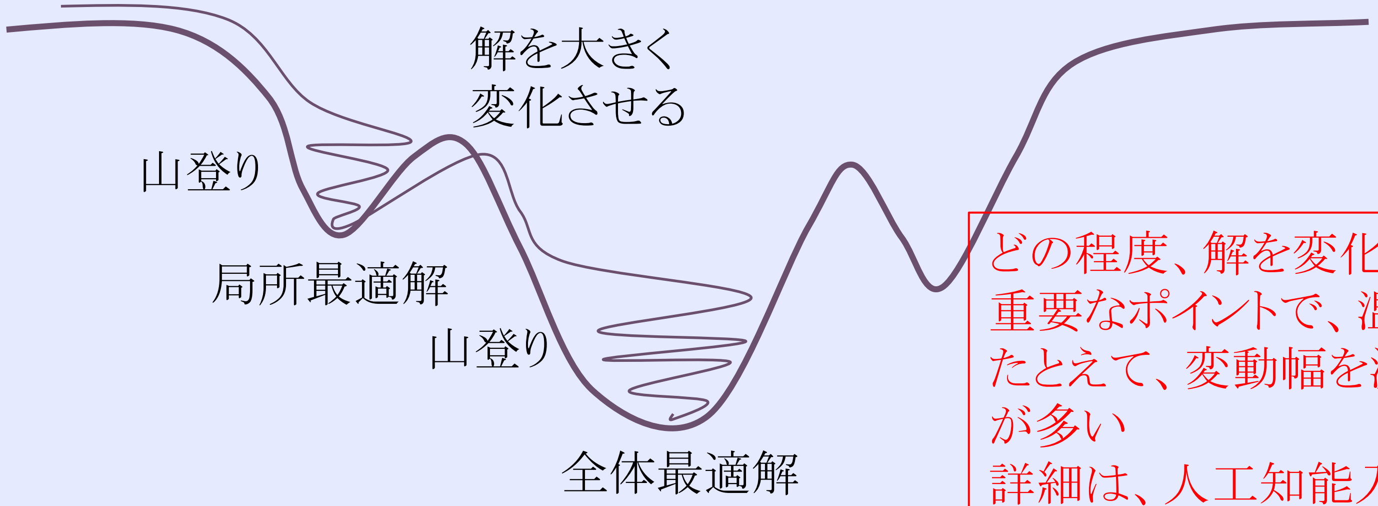
局所最適解と全体最適解

- 多峰性のある問題では、山登り法を実施すると局所最適解(local minimum)に到達してしまうことが多い
  - 局所解を脱して、全体最適解(global minimum)に到達する方法として考えられたのが焼きなまし法

焼きなまし
炉で刃物を熱し、それを徐々に冷ましながら打ち、再び炉で熱して、また打つということを繰り返して、刃物の強度を高める。

Simulated Annealing

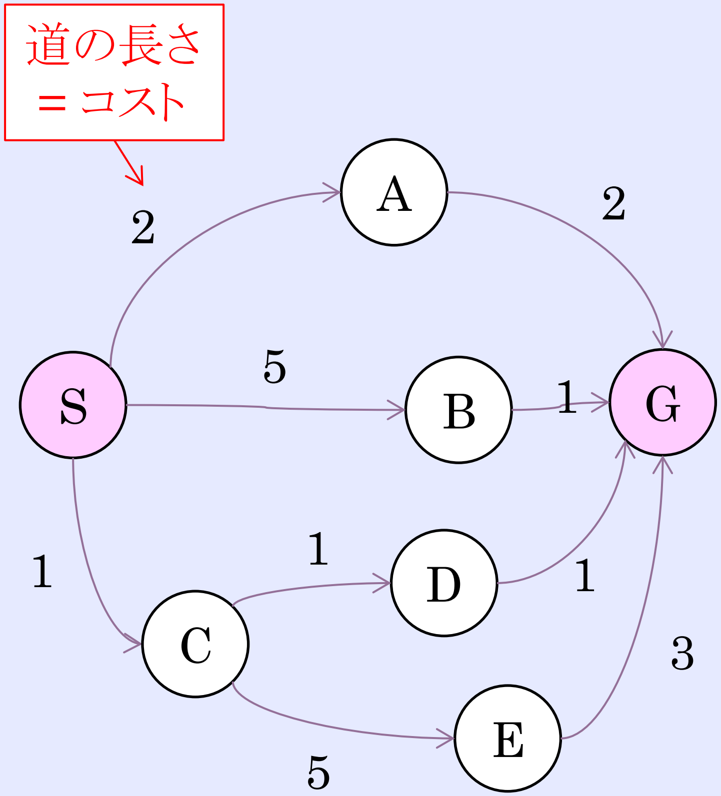
- 焼きなましを模擬して、最適解を探索する手法
  - 山登り法で局所最適解を探索する
  - 局所最適解から、一旦、解を大きく変動させる
  - 以上の2ステップを繰り返しつつ、変動幅を小さくする



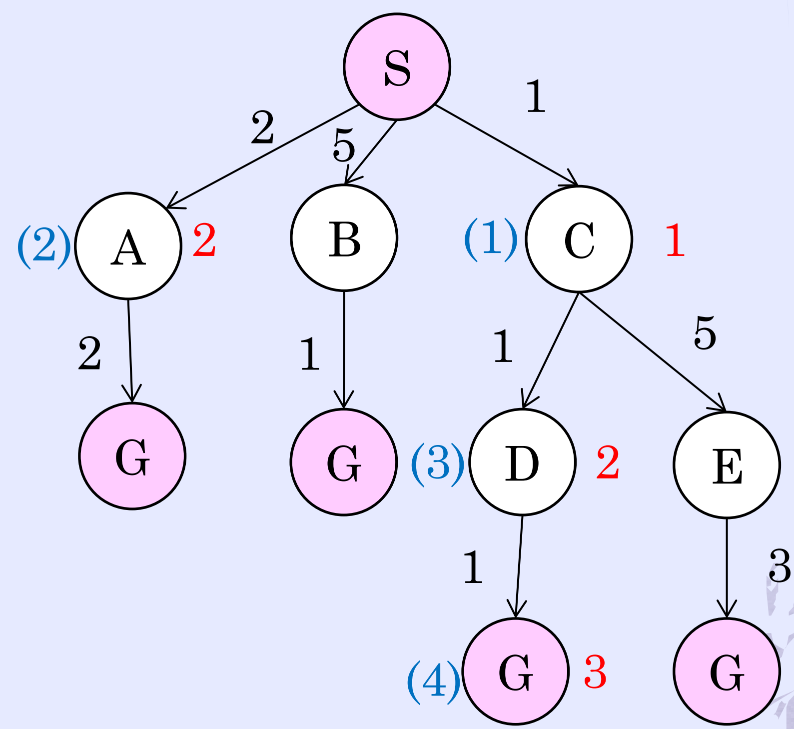
どの程度、解を変化させるかが重要なポイントで、温度制御にたとえて、変動幅を決めることが多い
詳細は、人工知能入門では扱わない

例: 経路探索

- 経路探索問題で最良優先探索で最短経路を求める



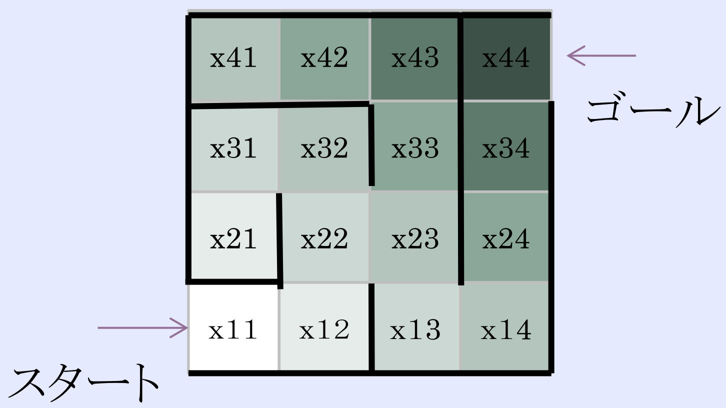
探索木



最良優先探索

(演習1)迷路

- 次の迷路の探索木を作成し、バックトラック付き山登り法と最良優先探索のそれぞれの探索パスを回答せよ。



A*アルゴリズム

A*（**A-star**）探索アルゴリズムは、グラフ探索アルゴリズムの一つ。最良優先探索を拡張したに、さらに**f**値として「現時点までの距離」**g**と「ゴールまでの推定値」**h**の和を採用したもの。**h** は ヒューリスティック関数と呼ばれる。

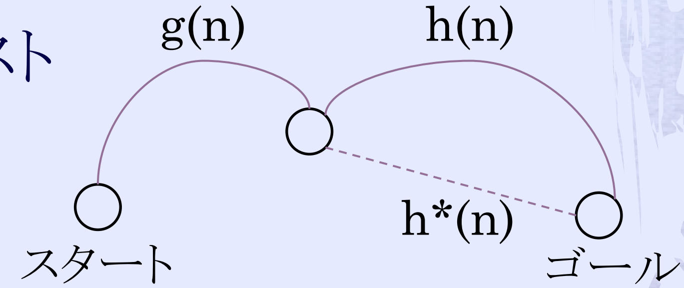
最短経路探索、再び

- 幅優先探索、繰り返し深化、最良解探索のいずれも、東小金井駅の南側の探索は無駄

- オープンリスト中のノード

ゴールまでのコスト

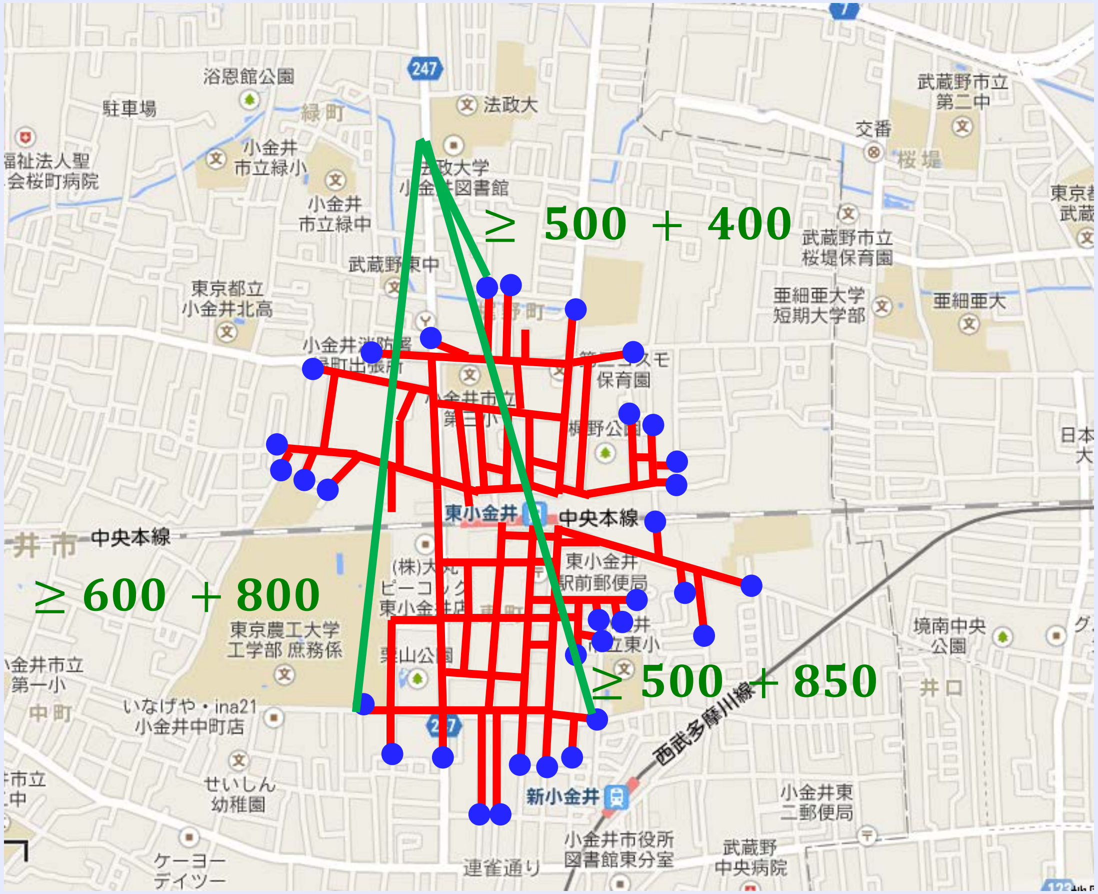
- ノードnにおいて、
  - f(n): ノードnを通って、スタートからゴールまでのコスト
  - g(n): スタートからノードnまでのコスト
  - h(n): ノードnからゴールまでのコスト
  - f(n) = g(n) + h(n)
- それぞれ予想値を * で表すと
  - f*(n) = g*(n) + h*(n)
    - g*(n)は探索中のノードnまでのコストなので既知
    - h*(n)は、探索しないとわからない。しかし、地図探索であれば、ゴールとの直線距離よりは長いことは明らか。
    - h(n)≥h*(n)≥0なるヒューリスティック関数h*(n)を用いると、f(n)≥f*(n)となり、ノードnを通るパスの最小予想値がわかる



f(n) = g(n) + h(n) ≥ g(n) + h*(n)

f*(n) を用いて最良優先探索を行う

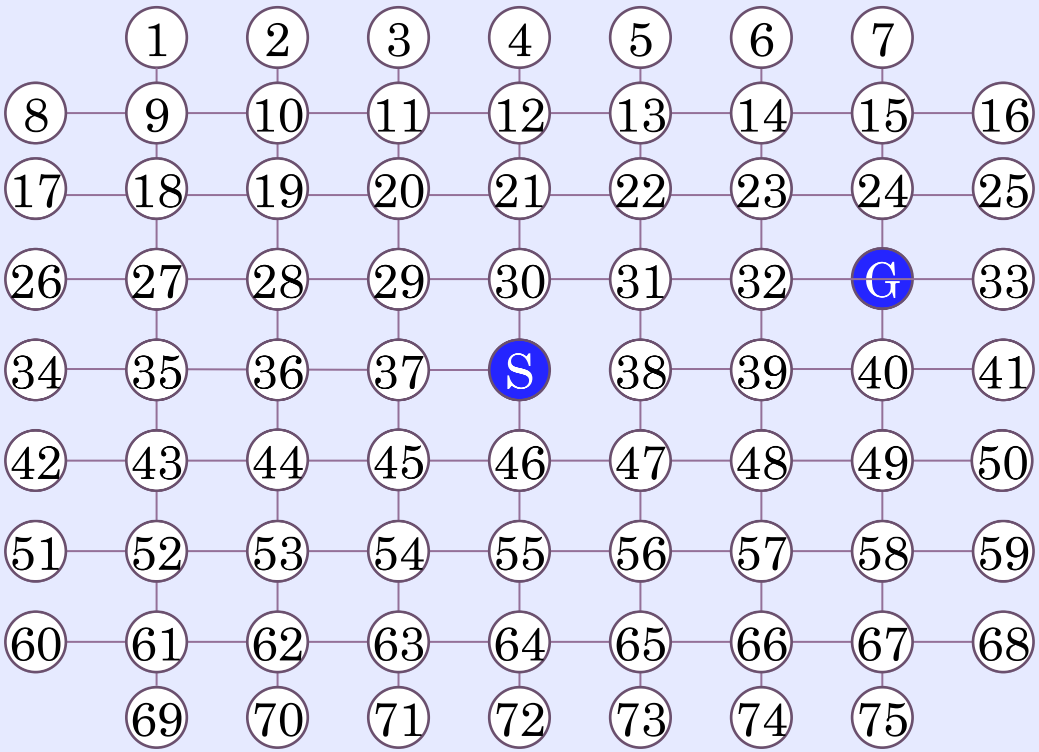
- 東小金井駅の南側のノードは、f*(n)が大きくなるので、探索の優先順位が下がる

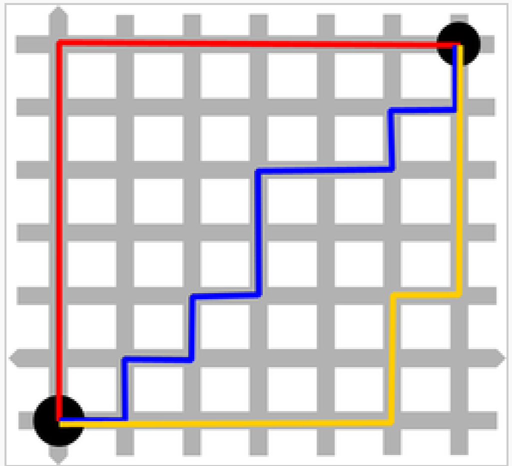


- オープンリスト中のノード

格子上の経路探索

- 経路が格子に限られている場合は、h*(n)としてマンハッタン距離を利用可能

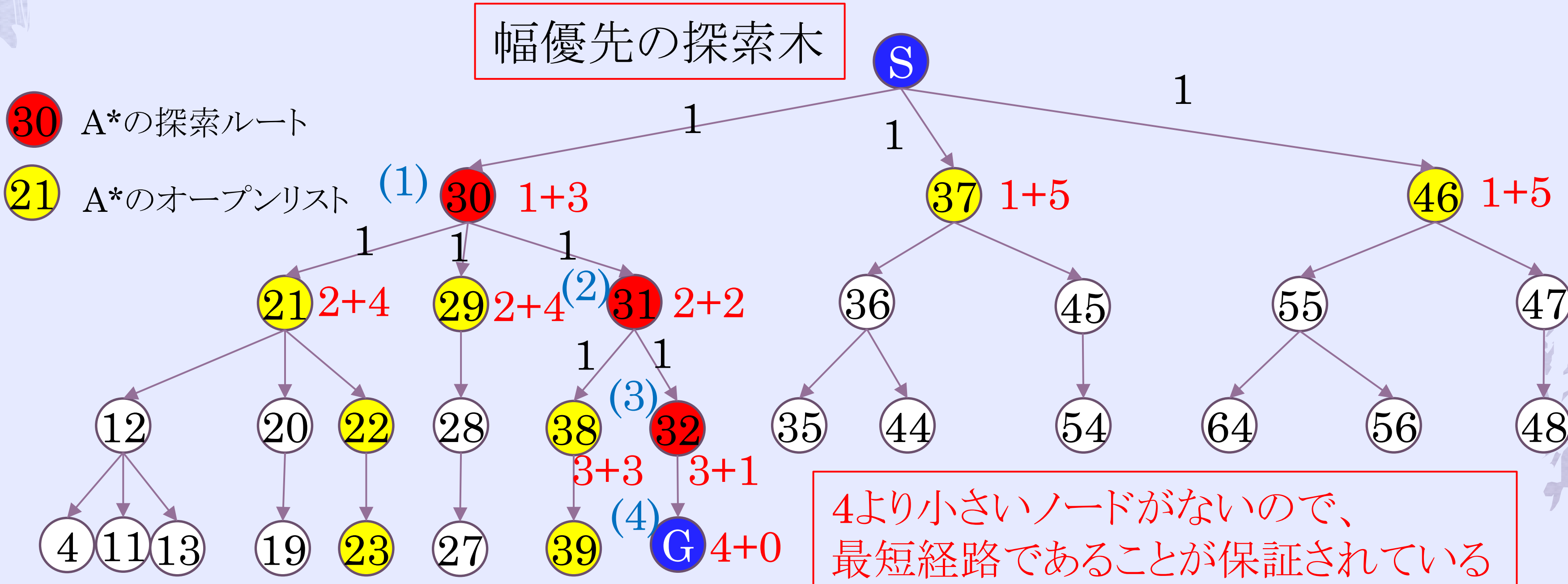




マンハッタン距離の例: どの色のコースを辿っても同じ距離が決まっている

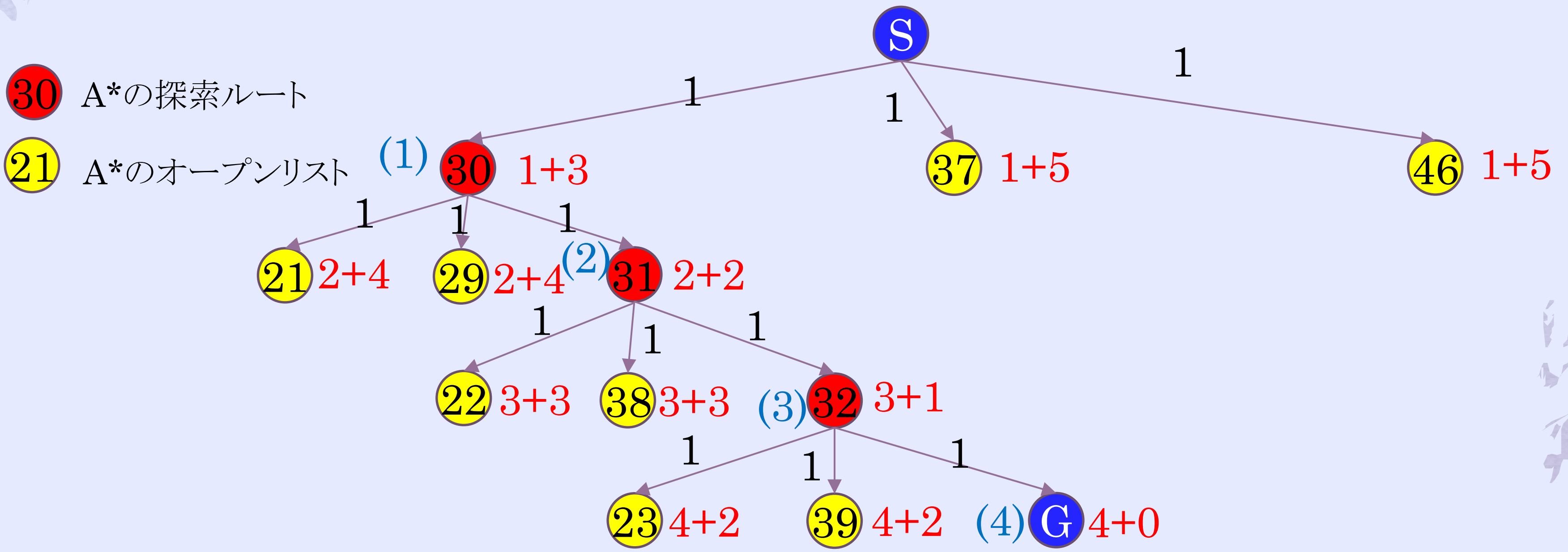
探索ノード数

- 幅優先探索
  - 探索ノード数　30
- Iterative Deeping
  - 探索ノード数　58
- A*アルゴリズム
  - 探索ノード数　13(なぜ13かは、次ページの図を参照のこと)



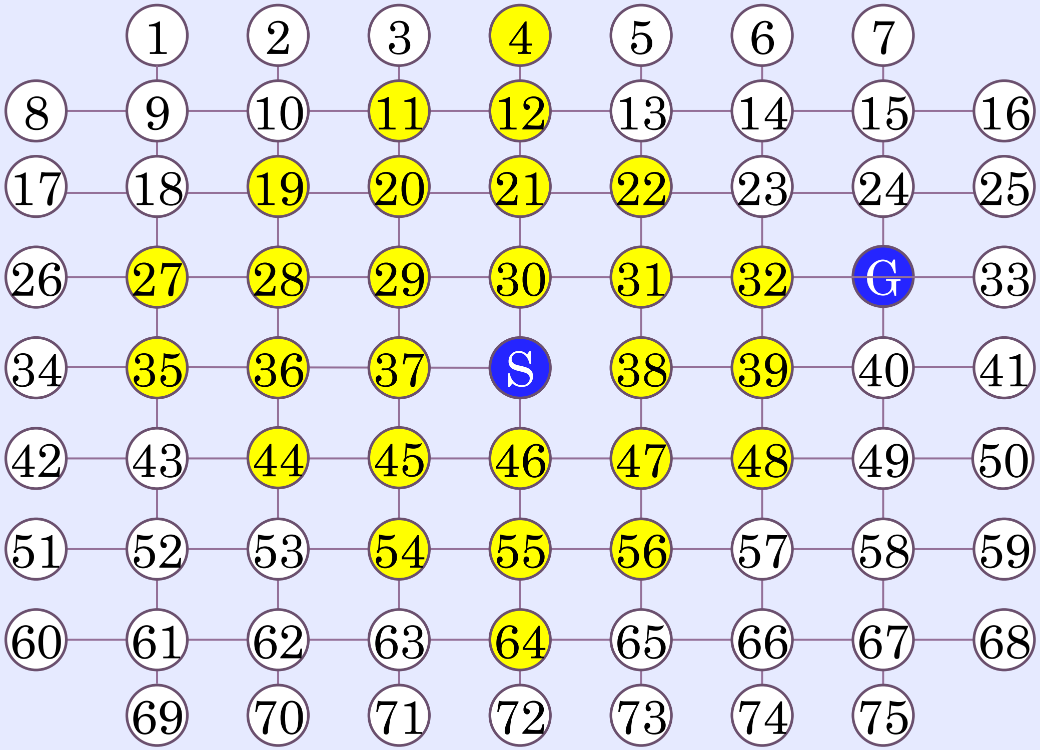
A*の探索木

- 幅優先探索
  - 探索ノード数　30
- Iterative Deeping
  - 探索ノード数　58
- A*アルゴリズム
  - 探索ノード数　13(なぜ13かは、次ページの図を参照のこと)



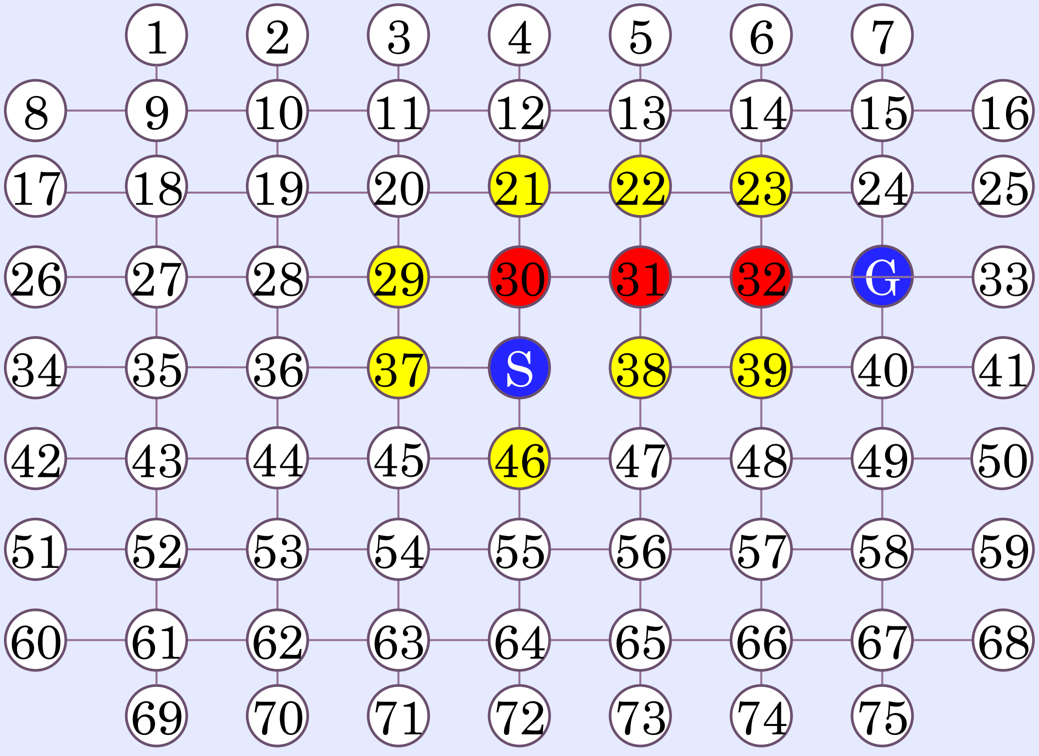
格子上の経路探索

- 幅優先探索の場合
  - スタート地点の周りに探索ノードが広がる



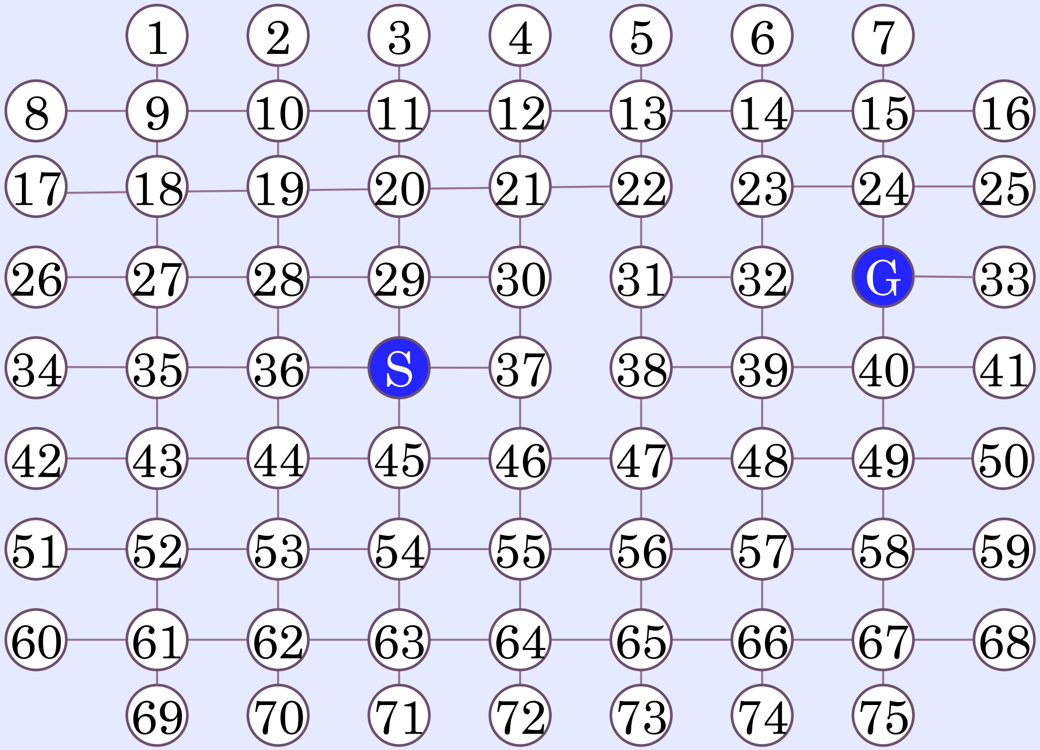
格子上の経路探索

- A*アルゴリズムの場合
  - スタート地点とゴール地点の間の最短経路を中心に探索範囲が縮まる。



(演習2)経路探索

- h*(n)としてマンハッタン距離を利用して、スタートからゴールまでの最短経路をA*アルゴリズムで求め。探索順は、上左下右とする。



Summary of Search Strategies

Strategy	Frontier Selection	Halts?	Space
Depth-first	Last node added	No	Linear
Breadth-first	First node added	Yes	Exp
Heuristic depth-first	Local min $h(n)$	No	Linear
Best-first	Global min $h(n)$	No	Exp
Lowest-cost-first	Minimal $cost(n)$	Yes	Exp
A^*	Minimal $f(n)$	Yes	Exp

(課題)

1. $f(x) = x^2 - x + 3$ の最小値を、$x = 0$ から始めて、0.2刻みで山登り法で探索し、動けなくなったところで、0.1刻みに変えて探索することで求めよ
2. 下記の経路探索問題で、h*(n)にユークリッド距離を用いる場合とマンハッタン距離を用いる場合でオープンリスト、探索経路の違いを調べよ。探索順は上左下右したとする。
3. (オプション: 加点) 課題2の経路探索問題をプログラミングして、実際に、A*アルゴリズムを動作させよ。

ユークリッド距離の例 :
d(17,13)= 1.4
d(17,18) = d(17,12) = 1
d(17,G) = 2.8
d(23,G) = 3.2