応用アルゴリズム演習
―動的計画法―

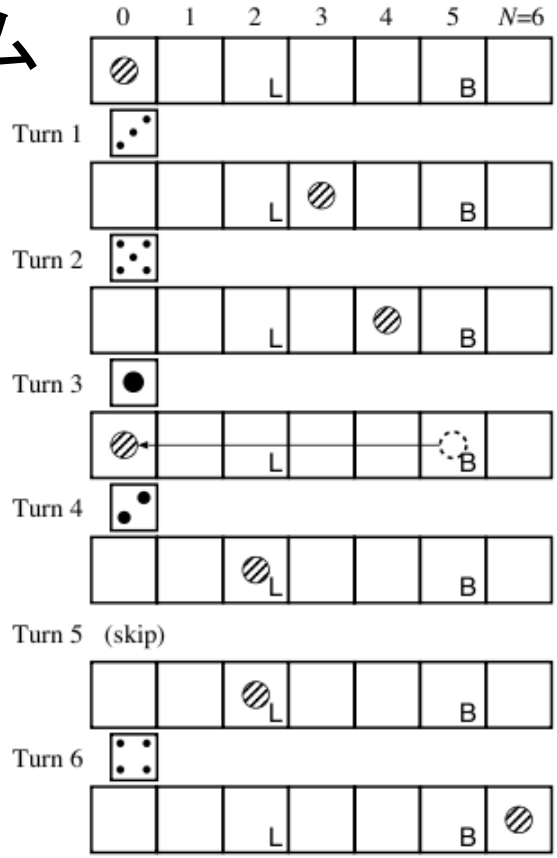
鎌田十三郎

本日の内容

- 動的計画法 (Dynamic Programming)
- 計算量と実際の量感覚

例題

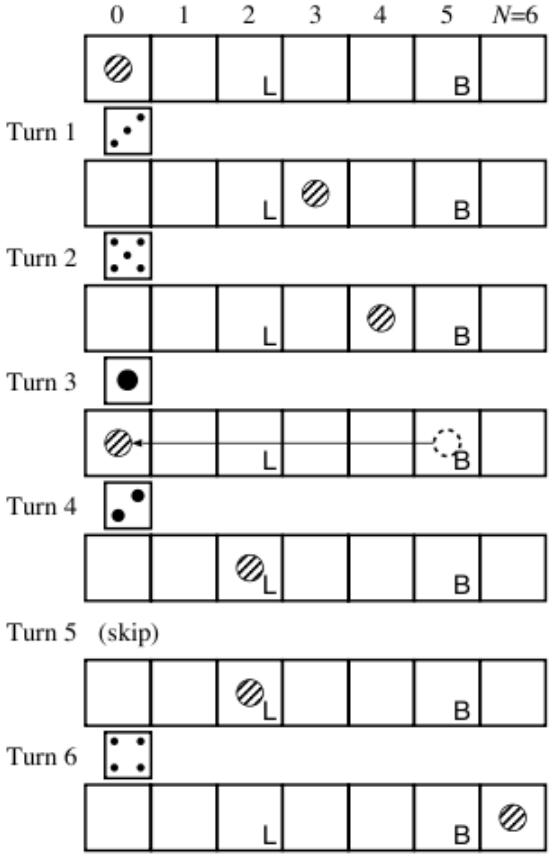
- バックギャモンを単純化したゲーム
  - 左端からスタート、右端がゴール
  - サイコロを振って、目の数だけ進む
    - ゴールを超えたら、その分戻る
  - L: 1回休み
  - B: スタートに戻る
- 質問： T 回以内にゴールにたどり着く確率は？(1≦T≦100)



どうやったら解けそう？

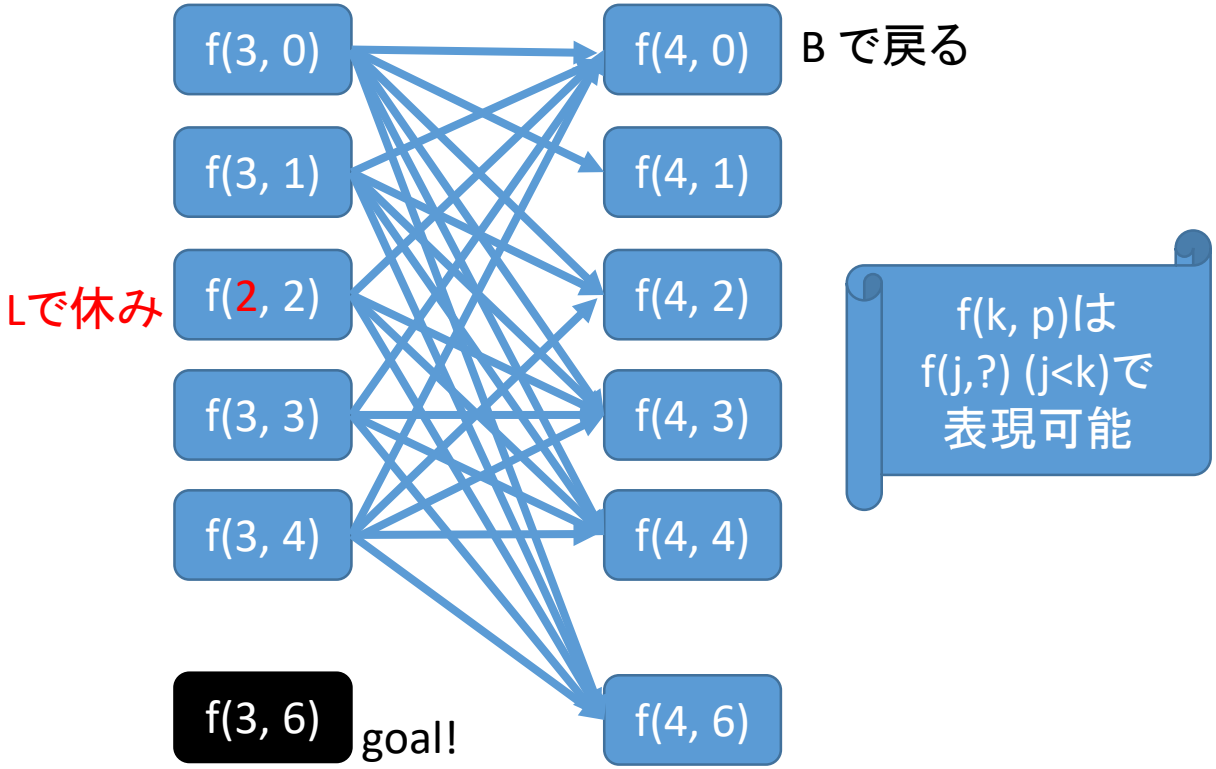
試行錯誤

- 最初は必ずスタート
- 1/6 の確率で 1 から 6 に移動
  - 但し、L についたら1回休み
  - B についたら、スタートに戻る
- サイコロのパターンを全部試す？
  - 10回振るパターン: 6^{10} ～60M
  - 20回だと、6^{20} ～ 3600ペタ？？
  - 100回なんて、、、、

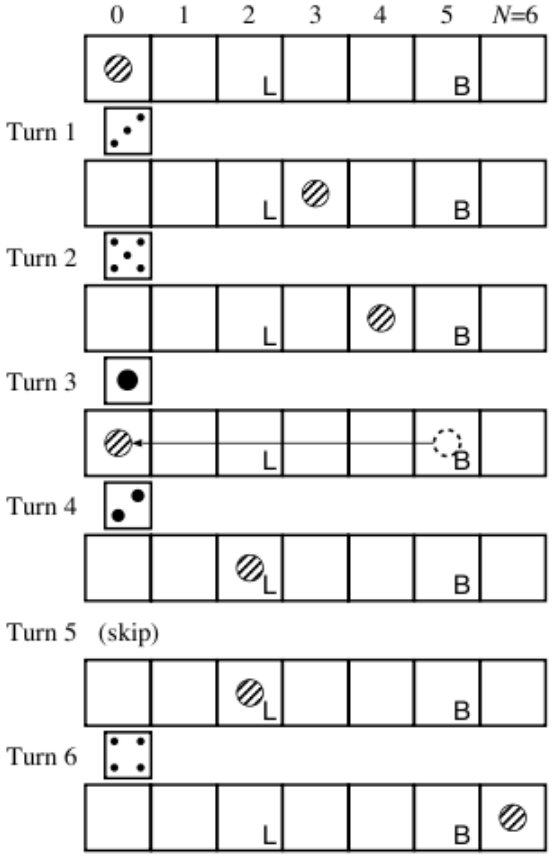


解法

- f(k, p): k回目に位置 p に到着する確率とする



f(k, p)は
f(j,?) (j<k)で
表現可能



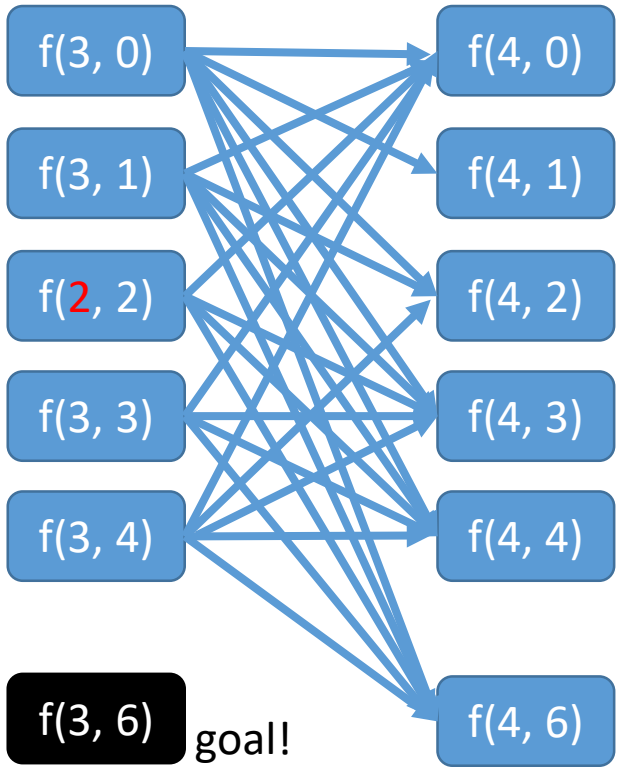
動的計画法 1/2 (Dynamic Programming, DP)

- 部分問題から順に解き、
- 部分問題の結果を用いて、より大規模な問題を解く

```
int prob[MAX_T][MAX_N+1];  /* f 相当 */

int solve(int n, int t) {
    prob[0][0], .., prob[0][n] の初期化
    for(k ….) {
        prob[j][?] (j<k)から prob[k][n]を求める
        もしくは
        prob[k][?] を prob[j][?] (j>k)に反映
    }
}
```

今回はこっちが楽かと？



動的計画法 2/2 (Dynamic Programming, DP)

- 実は、ダイクストラ法なども DP の一種
  - 波紋を広げながら、問題規模を大きく
- フィボナッチ関数を loop で解くのも、DP的解法
- プログラム：単純なループ構造になることも多い
  - ＋結果の記録用データ構造（配列 or HashMapなど）

計算量

- 今回の問題
  - 各ターンの計算：O(n)　(n: 盤面サイズ)
  - ターンの数:　t
  - 時間計算量: O(n*t)
  - 空間計算量：O(n*t)？　　実際には、O(n)で大丈夫。

  - 5≦n≦100,　1≦t≦100
  - 100 x 100 なんて一瞬さ！！
  - 100 x 100 の配列なんて、なんてことないぜ！

現実計算機のスペック

- CPU のクロック：数GHz
  - 1秒間に数G回の命令をこなす（1命令1nano sec以下）
  - 1000命令かかる処理でも、数M回こなす

但し、主メモリランダムアクセスは、CPUより数十倍遅い

- メモリ容量
  - 主メモリ数GBは当たり前
    - でも、OS の領域も残してね
  - ハードディスクはTBオーダー
    - でも、disk は遅いけどね

でも、キャッシュは数MB程度以下

量感覚イメージ

- 累乗: 2^{10}=K, 2^{20}=M, 2^{30}=G
- 階乗:
  - 1x2x3x4x5=120, 1x2x...x10=3.6Mぐらい
  - ..x13 で4G越える
- 32bit 符号付整数で表現できる値:　-2G ～ +2G
- 時間
  - 1秒に数G回の演算が可能
  - 1分:60秒、1時間:3.6K秒、1日:86.4K秒、
  - 1M秒で11日半ぐらい