SRM570 div1 mediumの解説（自分用）

◆問題文◆
- N台のサーバがある
- サーバは木構造のネットワークを構築
- それぞれのサーバを2組で分ける
- 2組で分け終わった後に、それぞれの組みでサーバのスロットをケーブルでつないで木構造のネットワークを構築
- サーバに新規スロットを新しく自由に設置することができる
- 2組の木ネットワークを作るのに必要な新規スロットの最小値の期待値を求める。

◆条件◆
- ケーブルは無限にある
- 最初から連結していた成分はそのまま連結している
- スロットはサーバごとに1つ既に備わっている（1つのサーバに新しく2つ以上のケーブルをつなぐとき新規スロットが必要になる）
- 分ける前のネットワーク用に使われていたスロットは再利用不可能

考察

◆木構造◆

- ノード数V　エッジ数E　とすると
  連結成分数：　1
  V = E + 1

- 連結成分が1つ増えると
  連結成分数：　２
  V = E + 1 + 1 = E + 2

- 連結成分数Cでは
  V = E + C

- 全体を木にするために必要なケーブルとスロットの数は
  ケーブル数：　C – 1
  スロット数：　2 * (C – 1)

- 最初からサーバ1台につきスロットが1つ備わっているので
  必要なスロットの数は
  スロット数：　max(2 * (C – 1) – V, 0)

考察（つづき）

- 新しいサーバを接続するときはどこにつないでもいい
- どこのサーバに接続しても木構造は崩れないので
  最初から備わっていたスロットを優先的に使わせる
- ノード数　エッジ数　連結成分数の3つだけで必要なスロットの数は求まる
  V = E + C を使えば　2つあれば求まる

  スロット数：　max(2 * (V – E – 1) – V, 0) = max(V – 2 * E – 2, 0)
- ノード数V エッジ数E になるようなサーバの選び方の場合の数をすべて求める
  量が多いからDPする
- 場合の数の総和はpow(2, V)　選び方は2つの組みで対照的なので
  片方について求めて2倍すればいい

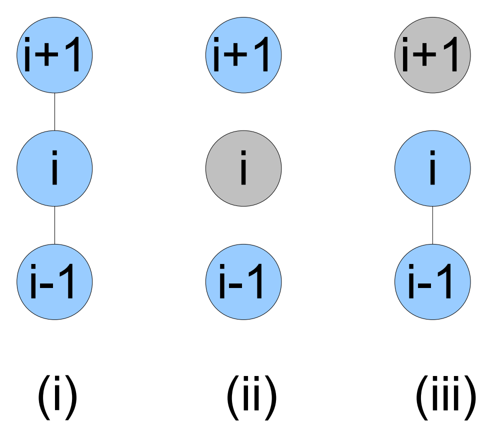
  よって答えは Σ{dp[v][e] * max(v–2*e–2, 0)}/pow(2, V-1)

木DP

- 根付き木の葉っぱから木DPする
  葉っぱから番号をつける(dfsを使う)
    dp[i][v][e] = i番目以降の子供のノードを使った場合の場合の数
  とし親ノードのテーブルを子ノードで更新していく
    dp[i+1][v][e] = f(dp[i][v][e], dp[i-1][v][e], ...)

  - dp[v][e] = dp[V][v][e] (根に集まったテーブル)

- i番目以降の子供のネットワークにi+1番目をつなげたときのv, eの変化が知りたい
  (i)i番目とi+1番目がつながる場合
    $v \to v+1,\ e \to e+1$
  (ii)i番目とi+1番目がつながらない場合
    $v \to v+1,\ e \to e$
  (iii)i+1番目を使わない場合
    $v \to v,\ e \to e$



親ノードと子ノードの関係

- 親ノードのテーブルを更新するとき、直近の子ノードがネットワークにあるかないかでv, eの量が変わる
  i番目のノードを使った場合(j = 1)と使わなかった場合(j = 0)で分けて計算する
    dp[i][v][e] → dp[i][j][v][e]
    に拡張

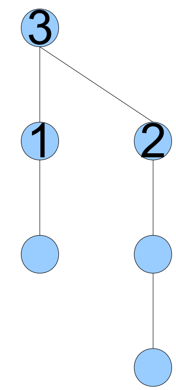
  i+1番目のノードをi番目以降の子ノードで更新
    (i)i+1番目を使う　i番目を使う
      dp[i+1][1][v+1][e+1] ← dp[i][1][v][e]

    (ii)i+1番目を使う　i番目を使わない
      dp[i+1][1][v+1][e] ← dp[i][0][v][e]

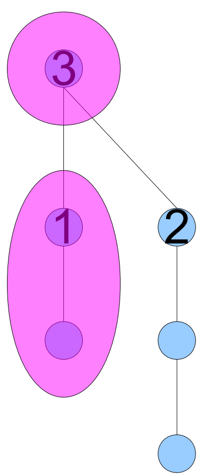
    (iii)i+1番目を使わない　i番目を使う
      dp[i+1][0][v][e] ← dp[i][1][v][e]

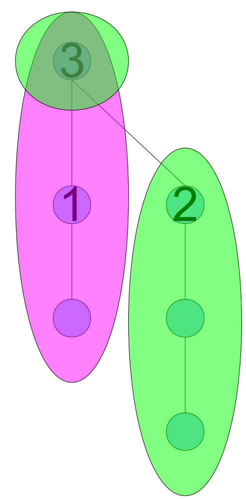
    (iv)i+1番目を使わない　i番目を使わない
      dp[i+1][0][v][e] ← dp[i][0][v][e]

子ノードが複数ある場合



- 子ノードが複数ある場合の更新
  (i)まず１つのノードを使って親ノードを更新
  (ii)他の子ノードについては、親ノードと子ノードの掛け算で更新する





dp[3][1][v+1][e+1] ← dp[1][1][v][e]
dp[3][1][v+1][e] ← dp[1][0][v][e]
dp[3][0][v][e] ← dp[1][1][v][e]
dp[3][0][v][e] ← dp[1][0][v][e]

dp[3][1][v+v'][e+e'+1] ← dp[3][1][v'][e']*dp[2][1][v][e]
dp[3][1][v+v'][e+e'] ← dp[3][1][v'][e']*dp[2][0][v][e]
dp[3][1][v+v'][e+e'] ← dp[3][0][v'][e']*dp[2][1][v][e]
dp[3][1][v+v'][e+e'] ← dp[3][0][v'][e']*dp[2][0][v][e]

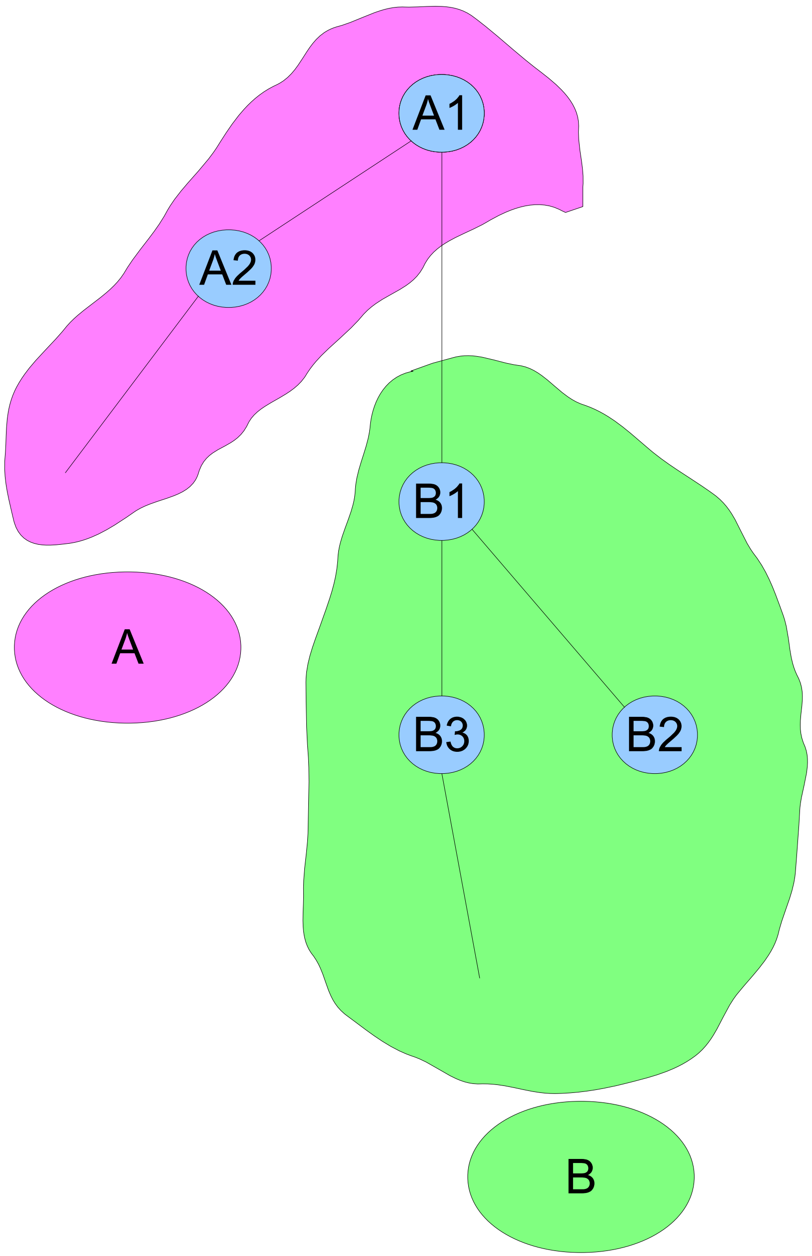
結局

- 子ノードが複数ある場合への一般化
  (i)親ノード単独の場合
  親ノードPについて
  dp[P][1][1][0] = 1
  dp[P][0][0][0] = 1
  で初期化
  (ii)子ノードを含めた場合
  - 子ノードcについて
    dp2[P][1][v+v'][e+e'+1] += dp[P][1][v][e]*dp[c][1][v'][e']
    dp2[P][1][v+v'][e+e'] += dp[P][1][v][e]*dp[c][0][v'][e']
    dp2[P][0][v+v'][e+e'] += dp[P][0][v][e]*dp[c][1][v'][e']
    dp2[P][0][v+v'][e+e'] += dp[P][0][v][e]*dp[c][0][v'][e']
  - 更新後の値で２重に更新しないように別テーブルdp2に保存して
    あとでまとめてdpテーブルに上書き
    [4][3] += [2][3]*[2][0]
    [5][4] += [4][3]*[1][1] ← ここでの[4][3]は += する前の値を使いたい
  - dp[P][j][v][e] == 0 のときはcontinue;
  - dp[c][j][v'][e'] == 0 のときもcontinue;
- (i) → Σ(ii) を親ノードについて葉から求めていく

補足

- 計算量
  親ノードと子ノードについて1回ずつ更新(エッジ数E)
  親ノードと子ノードのdpテーブルでループ((2*V*E)^2)
  - 計算量 = E*(2*V*E)^2 ~ V^5
    - 10^8 ~ 10^9 ぐらいできわどいが実際にはdp == 0でcontinueしているためもっと少ない
- 結果
  最大46ms

追加資料①



◆2つのネットワークをつなげた場合を考える
AとBはA1とB1の間でしか繋がっていない

(i)Aのネットワークのうち”A1を使わずに”
　　ノード数v=2, エッジ数e=1
となるネットワークの場合の数が3あるとする

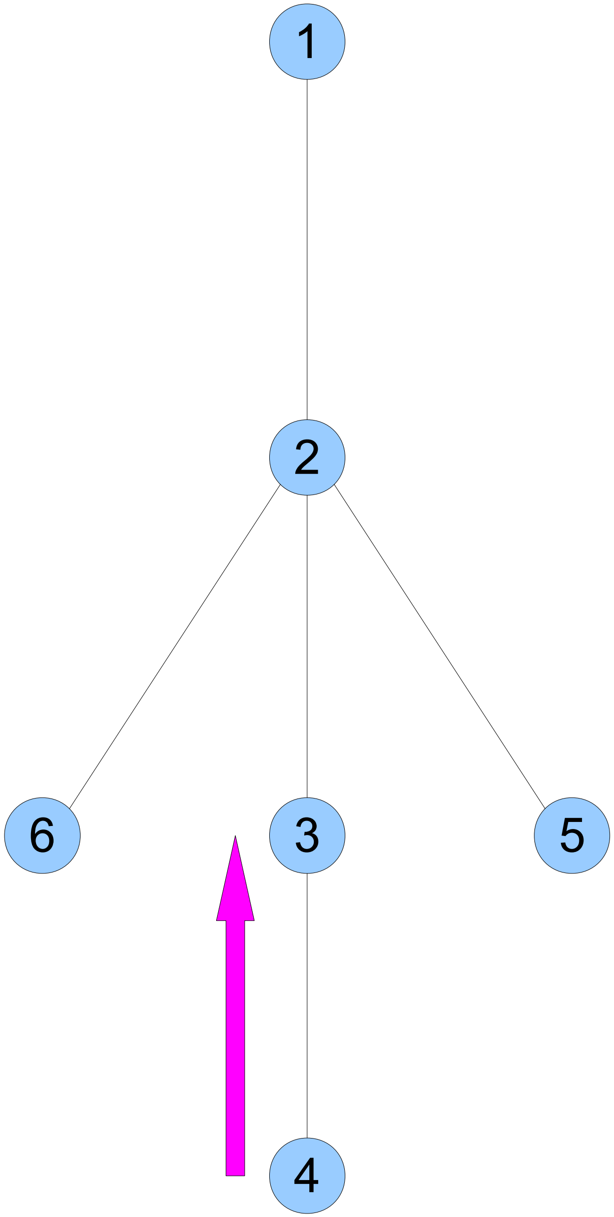
(ii)Bのネットワークのうち
　　ノード数v=4, エッジ数e=2
となるネットワークの場合の数が4あるとする

◆A1を使わなければ、AとBがつながることはありえないのでA、Bのネットワークを合体させてもノード数、エッジ数ともに変化はない（単純和）
つまり
　　v=2+4=6, e=1+2=3
となる組み合わせが
　　3*4=12通り出来上がる

　　v=6, e=3となる組み合わせは他にも
　　(v,e)=(3+3, 1+2), (5+1, 3+0),...
　　などある

◆A1とB1を両方使うならA1とB1の間のエッジをカウントするから
　　v=v(A)+v(B), e=e(A)+e(B)+1
になる

追加資料②



葉は初期化だけ
選ぶか選ばない
かなので2通り
[4][0][0][0] = 1
[4][1][1][0] = 1
Otherwise = 0

親ノード初期化
２通り
[3][0][0][0] = 1
[3][1][1][0] = 1
Otherwise = 0

親ノードを子ノードで更新
2^2 = 4通り
[3][0][0][0] = [3][0][0][0] * [4][0][0][0] = 1
[3][0][1][0] = [3][0][0][0] * [4][1][1][0] = 1
[3][1][2][0+1] = [3][1][1][0] * [4][1][1][0] = 1
[3][1][1][0] = [3][1][1][0] * [4][0][0][0] = 1