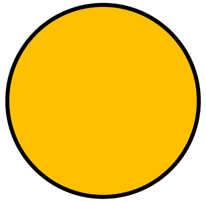
命綱

原案:itigo

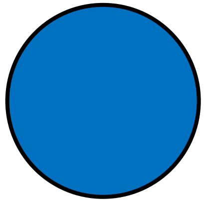
前提条件

- m回命綱がゴールに渡るためにはm-1回命綱が返ってくる必要がある。
- このm-1回分の命綱を返す作業を行うためにはm-1人ゴールから帰ってくる人を必要とし、以降のスライドでこの「帰ってくる予定のある人」と「二度と帰ってくる予定のない人」を以下のように塗り分ける
- 尚、「返ってくる予定のある人」はm-1人、「二度と帰ってくる予定のない人」はn人である。
- 以降スライドにおいて「帰ってくる予定のある人」は橙丸、「二度と帰ってくる予定のない人」は青丸と呼ぶ

帰ってくる予定のある人

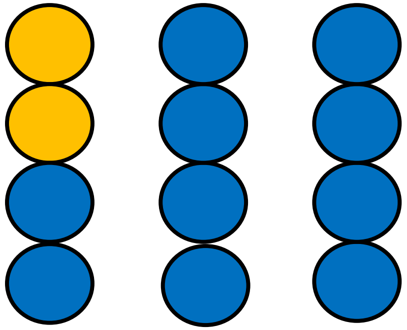


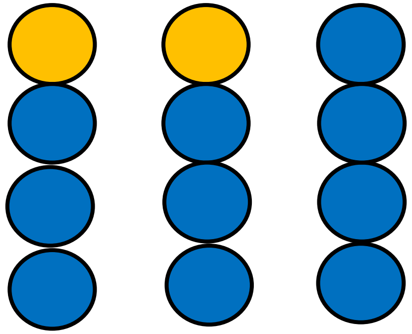
二度と帰ってくる予定のない人



例題

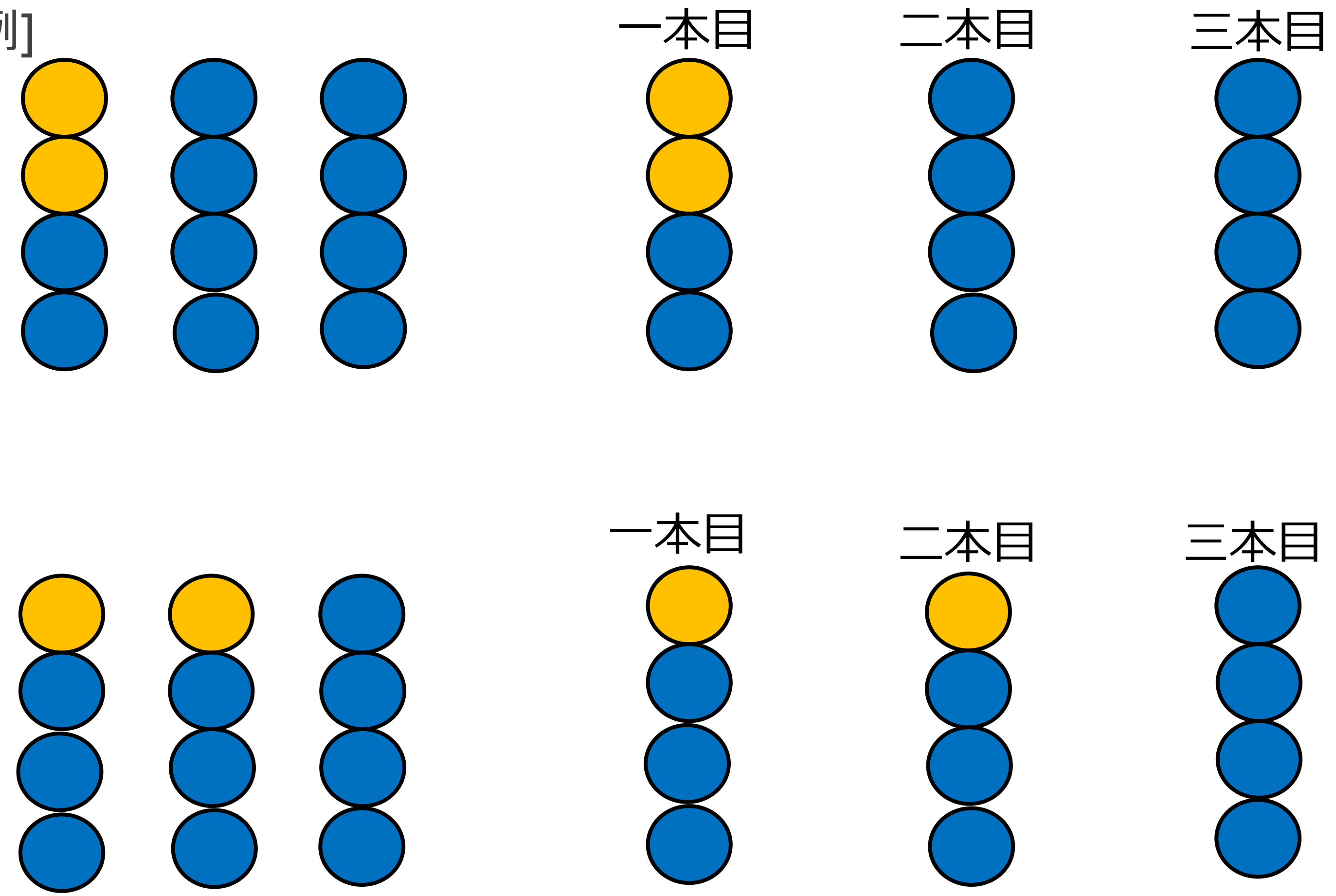
- 例えばn=10,k=4のサンプルに対し、m=3回命綱が渡ると仮定すると以下の二種類の命綱の構成員が考えられる。





- 橙丸の数がm-1ならば　橙丸をゴールに運ぶ→ゴールの橙丸がなくなるまで青丸のみの命綱を運ぶ　をくり返すことで任意の構成を運ぶことが可能である

[例]



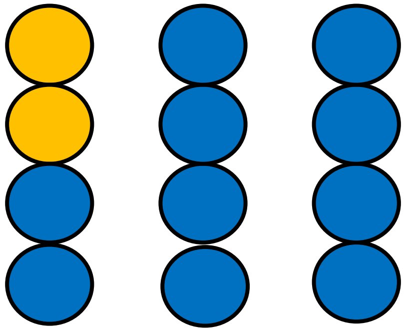
- よって本問題は以下の二つについて考察すればよい

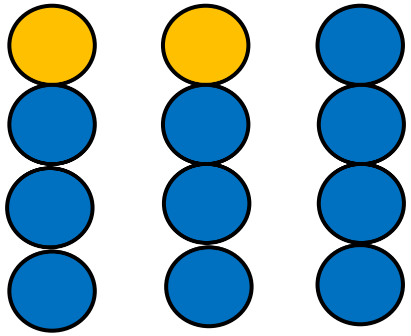
① 橙丸の配置が決定したとき、具体的な人間の割り当てをどのように決定するか

② 橙丸の配置をどのように決定するか

①、橙丸の選出

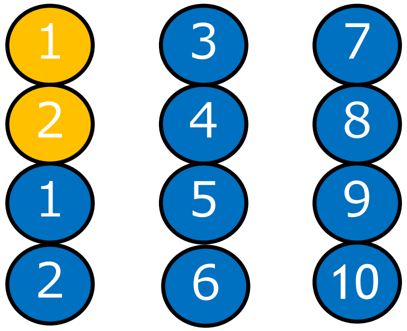
- 以降n=10,k=4,A={1,2,3,4,5,6,7,8,9,10}のケースについて考察する。
- このケースの場合以下の2通りの配置が考えられた

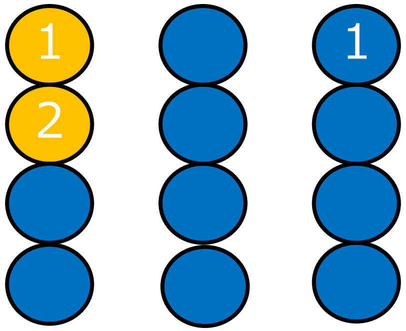




- ここで橙丸として誰を選出に関しては貪欲にコストが小さい人を選べばよいことが分かる(ゴールの橙丸が正の人数の時、 橙丸が居ない命綱を渡して対岸の橙丸を一人帰還させる。対岸の橙丸が0人の時、橙丸を含む命綱を渡す。を繰り返すと橙丸の選出を貪欲にコストが小さい人を選ぶことが可能)

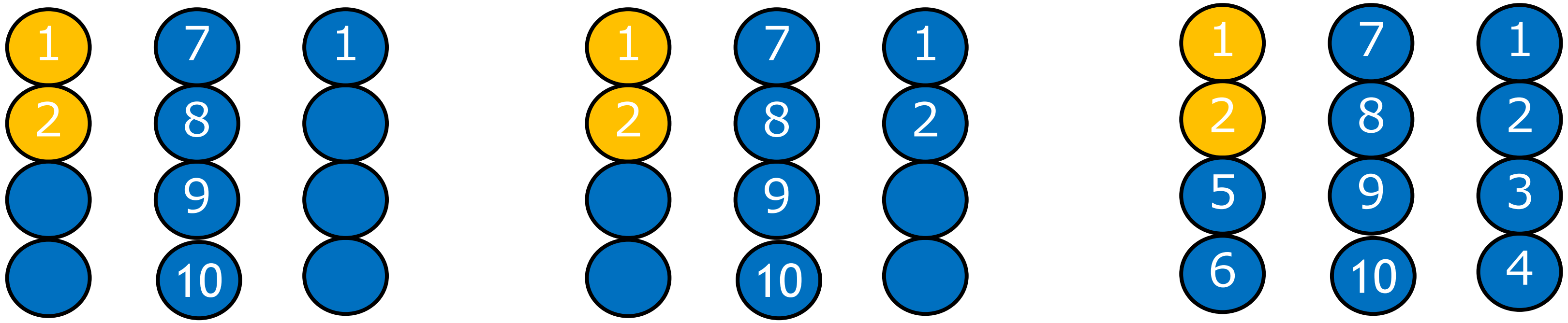
①、青丸の選出





- 青丸の選出方法であるが、上記左図のように選出してはいけない。なぜならば一本目の命綱において1,2番目の人が二重に選出されているからである。

- このように、橙丸として選出された人は自身の橙丸としての仕事を終えた後に選出される必要がある。
- 例えば、1番コストが軽い人については全て橙丸が移動し終えた後の最後の命綱に選出されることが確定する。(この情報は以降重要になる)



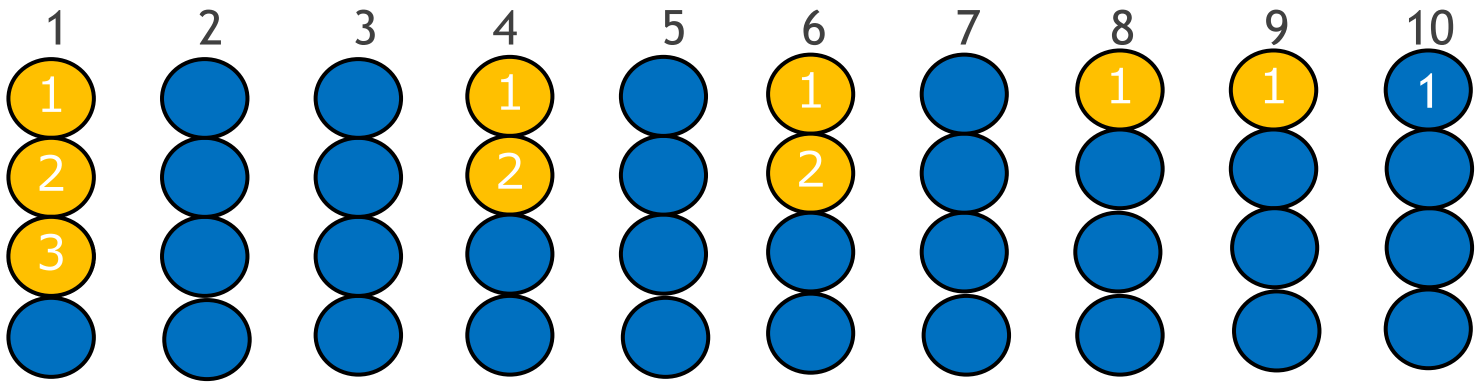
- さて、{2,3,4,5,6,7,8,9,10}の位置について考える。

- 命綱を運ぶコストはその命綱を構成する人の中で最もコストが大きい人のためこれを最小化するために貪欲に埋めていくことができる(具体的には一番余っている青丸が多い命綱を選択、その青丸にコストが重い順に人を埋める。を繰り返す(上図参照))

- 尚、このように埋めていった時、一度でも橙丸として選出された人は最後に運ばれる予定の命綱(前ページで1番コストが低い人が入った命綱)に入れて問題ないことが分かる(証明は次ページ)

証明1

- 命綱を橙丸が多い綱から順に消費するとする(以下は例)



上記例ではまず橙丸として選出される人、最後の命綱の1人が1番コストが低い人となることはこれまでの考察でわかる

さて6本目の命綱を運び終えた後、すなわち2番目にコストが低い人が橙丸としての仕事を終えたとき、どこに青丸として入るかを考察する。

証明2

この時、2番目にコストが低い人はまだ青丸として選出されていない人の中で1番コストが小さいため、残りの青丸の数が一番少ない命綱に選出するのがよいことが分かる

前ページ例の場合8,9,10番目の命綱のどれかである。

このうちどの命綱を選んでも命綱の残りの青丸の構成は同じため問題ない。

簡単のため10番目(最後の命綱)に入れて問題ない(N-1≡0 mod K-1よりこれが可能)

3,4,...番目にコストが小さい橙丸として選出されたことがある人も同様に最後の命綱で運ぶこととして問題ない

①、まとめ

- よって①は以下の手順で求めることができる

1.コストが軽い順に貪欲に橙丸を決定

2.橙丸として選出されたことがある人を全て青丸の命綱(これが結局最後に運ばれる命綱になる)の青丸に埋める

3.残りの人を青丸の余りが多い命綱から順に貪欲に埋めていく。

②、m=(n-1)/(k-1)は最適か？

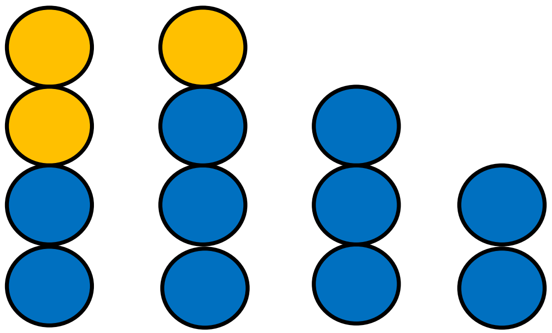
- 橙丸の並べ方について考えるが先に以下について考える。
- 命綱で運ぶ回数は可能な限り少ない回数になるよう可能な限り人を詰め込むことが最適か？
- 上記が正しいなら(n-1)%(k-1)=0の制約より命綱はk人の命綱が(n-1)/(k-1)個発生することとなる
- 結論から言うと以下のようになる

・全て命綱に青丸が存在するならm=(n-1)/(k-1)が最適

・橙丸だけの命綱が存在するならその限りではない

全て命綱に青丸が含まれるとき

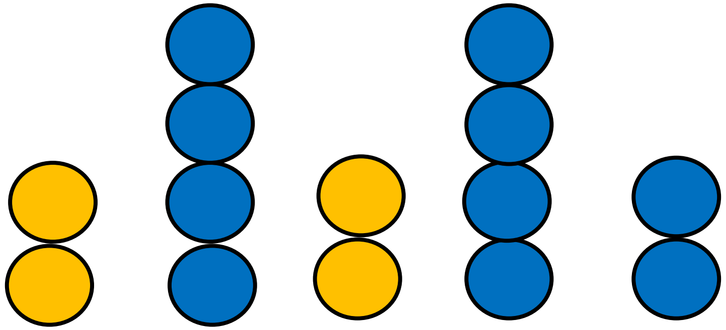
- 背理法で示す
- 例えばn=10,k=4で以下のような命綱の構成が最適と仮定する



- この時命綱のコストは構成員の最大コストであるため命綱を減らしてその構成員をほかの命綱に移動、交換するという操作を行うことにより必ず全体のコストを小さくできる
- 結局、全て命綱に青丸がある場合、可能な限り青丸が含まれる命綱を少なくすることが最適となる。

橙丸のみの命綱がある場合

- 反例としてn=10,k=4,A={1,1,1000,1000,1000,1000,1000,1000,1000,1000}が挙げられる。この場合以下がコスト2007となり最適になる



- 以上より先に命綱の青丸をどのように配置するかを決定し、余ってる場所、及び橙丸のみの命綱を作ることによって命綱の構成の決定とコストの計算が可能である

- これはdpを用いて計算できる。dpの書き方は様々考えられる。次ページ以降はその一例を示す

②、dpを用いて求める

- コストが重い人から順に青丸と橙丸の合計がk個の命綱を作っていく形で以下のdpを考える。(わかりにくい場合次ページへ)

dp[i][j][l]=コスト

i=何本目の命綱か

j=これまでに作った橙丸の数

l=前回の命綱の橙丸の数(コストが重い人は青丸が多い命綱に入る性質よりiが増加すると単調増加になる)

また遷移は以下のようになる

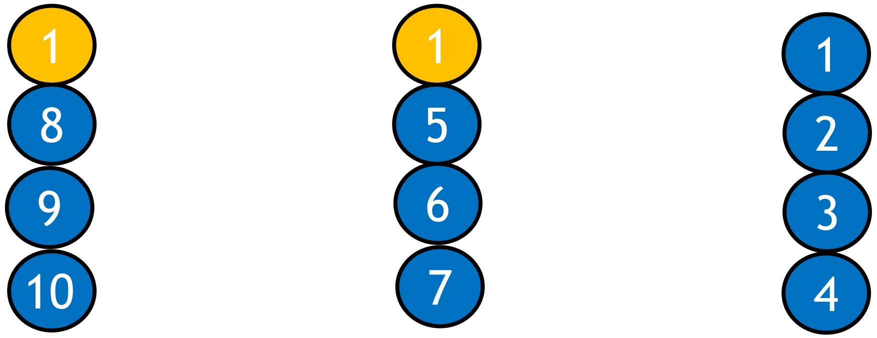
dp[i][j][l]=min($\bigcup_a dp[i-1][j-a][\,a] + (i-1$個目の命綱を渡すコスト$) + ($橙丸が帰還するコスト))

尚、最後の命綱だけ特殊な命綱になるため特別な扱いをする必要がある

dp例

- 前ページだけではわかりにくいのでdpの例を示す

dp[0][0][0]=0　dp[1][1][1]=11　dp[2][2][1]=19　dp[3][2][1]=23



dp[0][0][0]:命綱は無いのでコストは0

dp[1][1][1]:橙丸1つの命綱を作る。コストは青丸の最大値+橙丸の合計で11

dp[2][2][1]:さらに橙丸1つの命綱を作る。尚dp[1][1][1]からdp[2][1][0]に遷移するようなことはできない(追記:この遷移を許しても答えは変わらない)

dp[3][2][1]:i=(n-1)/(k-1)であることより橙丸として選出された人が乗せられた命綱であり、特殊な処理が必要。

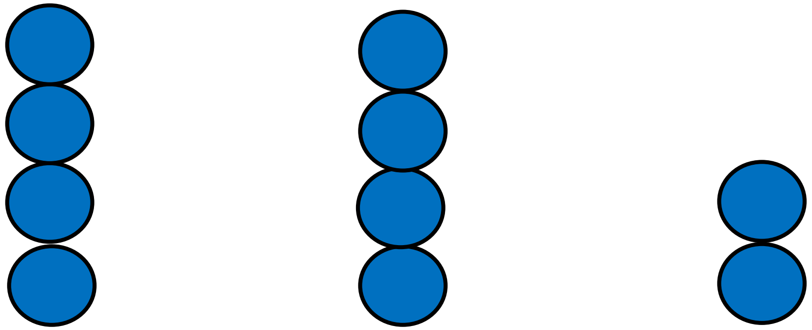
dpの注意点

- j>(n-1)/(k-1)-1には遷移しない(必要な命綱の本数以上の橙丸は必要ない)
- 青丸として選出される人と橙丸として選出される人が被る時、残りの人を全て青丸に埋める命綱を作る(橙丸に選出された人も含む最後に運ばれる予定の命綱)(この時全て人を青丸に埋めることが可能)
- この後、不足した橙丸を橙丸のみで構成された命綱を作ることにより補う
- lが減少する方向に遷移する必要はない

橙丸が不足したまま青丸が埋まった場合

- 例えば以下のようにdpが遷移した場合を考える。(スライド12pの例)
- n=10,k=4, A={1,1,1000,1000,1000,1000,1000,1000,1000,1000}

dp[1][0][0]=1000　dp[2][0][0]=2000　dp[3][0][0]=2001



この場合橙丸のみで構成された命綱によってあと二つの橙丸を対岸に運ぶ必要がある。(具体的には12pのように橙丸2つの命綱を二つ用意する。)

u個の橙丸を対岸に運ぶために必要な最小コストは簡単なdpで前計算が可能である。

計算量

- dpテーブルはiが(n-1)/(k-1)(命綱の本数)、jが(n-1)/(k-1)、lがkであり、遷移がkであるためO(n^2)で計算可能

- 尚、実際にこのdpを書くとMLEする可能性がある。この問題はdpの一次元目を無くし遷移を二つのdpテーブルで管理することにより対応可能である