問３　探索アルゴリズムに関する次の記述を読んで，設問 1～5 に答えよ。

1 個ずつの重さが異なる商品を組み合わせ，合計の重さが指定された値になるようにしたい。

この問題を次のように簡略化し，解法を考える。

〔問題〕
指定された n 個の異なる数（自然数）の中から任意の個数の数を選択し，それらの合計が指定された目標 X に最も近くなる数の組合せを 1 組選択する。その際，合計は X より大きくても小さくてもよい。ただし，同じ数は 1 回しか選択できないものとする。

例えば，指定された n 個の数が（10，34，55，77），目標 X が 100 とすると，選択した数の組合せは（10，34，55），選択した数の合計（以下，合計という）は 99 となる。

この問題を解くためのアルゴリズムを考える。

指定された n 個の数の中から任意の個数を選択することから，各数に対して，選択する，選択しない，の二つのケースがある。数を一つずつ調べて，次の数がなくなるまで“選択する”，“選択しない”の分岐を繰り返すことで，任意の個数を選択する全ての組合せを網羅できる。この場合分けを図 1 に示す。

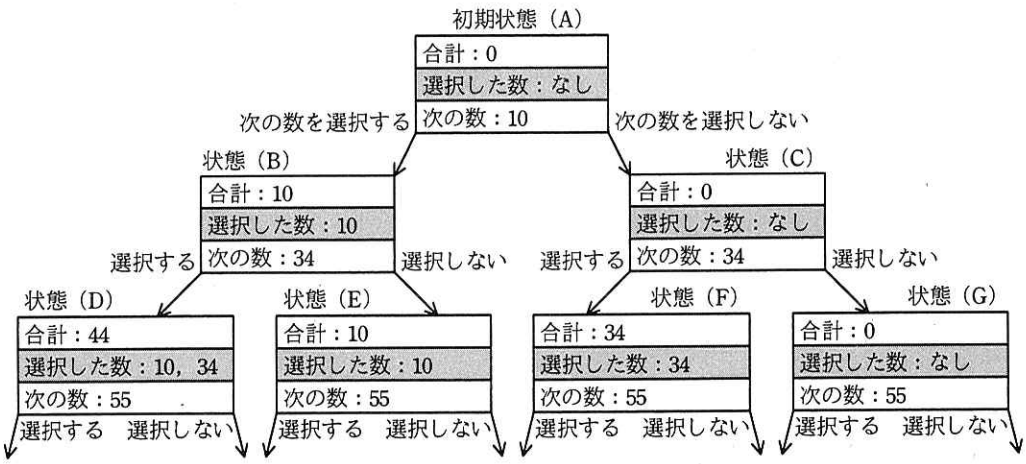

図 1　問題を解くための場合分け

〔データ構造の検討〕

図 1 の場合分けをプログラムで実装するために，必要となるデータ構造を検討する。

まず，図 1 の場合分けを木構造とみなしたときの各ノード（状態）を構造体 Status で表す。構造体 Status は要素として“合計”，“選択した数”，“次の数”をもつ必要がある。

プログラムで使用する配列，変数及び構造体を表 1 に示す。

表 1　プログラムで使用する配列，変数及び構造体

名称	種類	内容
numbers[ ]	配列	問題で指定される n 個の数を格納する配列。配列の添字は 1 から始まる。
target	変数	問題で指定される目標 X を格納する変数。
Status	構造体	次の三つの要素をもつ構造体。状態を表す。 ・total：合計を表す変数。初期値は 0。 ・selectedNumbers[ ]：選択した数を表す配列。各要素の初期値は null とする。配列の添字は 1 から始まる。 ・nextIndex：次の数の numbers[ ] における添字を格納する変数。初期値は 1。次の数がない場合は 0。 構造体の要素は“.”を使った表記で表す。“.”の左に，構造体全体を表す変数を書き，“.”の右に，要素名を書く。
currentStatus	変数	構造体 Status の値を格納する変数。“取得した状態”を表す。
ansStatus	変数	構造体 Status の値を格納する変数。“現時点での解答の候補”（以下，“解答の候補”という）を表す。初期値は null とする。

〔探索の手順〕

図 1 に示した場合分けの初期状態（A）からの探索手順を，次の (1)～(3) に示す。①これから探索する状態を格納しておくためのデータ構造として，キューを使用する場合とスタックを使用する場合で，探索の順序が異なる。また，②データ構造によってメモリの使用量も異なる。ここではキューを使用することにする。

(1) 初期状態（A）を作成し，キューに格納する。キューが空になるまで (2)，(3) を繰り返す。

(2) キューに格納されている状態を一つ取り出す。これを“取得した状態”と呼ぶ。“取得した状態”の評価を行う（状態を評価する手順は次の〔“取得した状態”の評価〕に示す）。

(3) “取得した状態”に次の数がある場合，次の数を選択した状態と，次の数を選択しない状態をそれぞれ作成し，順にキューに格納する。

〔“取得した状態”の評価〕

“取得した状態”を評価し，“解答の候補”を設定する手順を，次の(1)，(2)に示す。

(1) “解答の候補”がnullの場合，“取得した状態”を“解答の候補”にする。

(2) “解答の候補”がnullでない場合，“解答の候補”の合計と“取得した状態”の合計をそれぞれ目標Xと比較して，後者の方が目標Xに近い場合，“取得した状態”を“解答の候補”にする。

探索の手順が終了した時点の“解答の候補”を解答とする。

探索を行うための関数を表2に示す。

表2　探索を行うための関数

名称	内容
enqueue(s)	引数として与えられる構造体Statusの値sをキューに追加する。
dequeue( )	キューから構造体Statusの値を取り出して返す。
isEmpty( )	キューが空かどうかを判定する。 キューが空ならば1を，そうでなければ0を返す。
nextStatus1(s)	引数として与えられる構造体Statusの値sに対して，次の数を選択した状態を表す構造体Statusの値を返す。戻り値の各要素に次の内容を設定する。 ・total：s.total+numbers[s.nextIndex]を設定する。 ・selectedNumbers[ ]：s.selectedNumbers[ ]にnumbers[s.nextIndex]を追加した配列を設定する。 ・nextIndex：s.nextIndexがnならば0を，そうでなければs.nextIndex+1を設定する。
nextStatus2(s)	引数として与えられる構造体Statusの値sに対して，次の数を選択しない状態を表す構造体Statusの値を返す。戻り値の各要素に次の内容を設定する。 ・total：s.totalを設定する。 ・selectedNumbers[ ]：s.selectedNumbers[ ]を設定する。 ・nextIndex：s.nextIndexがnならば0を，そうでなければs.nextIndex+1を設定する。
abs(n)	引数として与えられる数nの絶対値を返す。

〔探索処理関数 treeSearch〕

探索処理を実装した関数 treeSearch のプログラムを図 2 に示す。

ここで，表 1 で定義した配列及び変数は，グローバル変数とする。

```
function treeSearch( )
  currentStatus を初期化する                    //初期状態を作成する
  enqueue( currentStatus )                      //初期状態をキューに格納する
  while( [        ア        ] )
    currentStatus ← dequeue()                   //キューから状態を取り出す
    if( ansStatus が null である )
      [        イ        ]
    elseif( abs(target-ansStatus.total)が abs(target-currentStatus.total)よりも大きい )
      [        ウ        ]
    endif                                       ←(α)
    if( [        エ        ] )                 ←(β)
      enqueue( nextStatus1(currentStatus) )     //次の数を選択した状態をキューに追加する
      enqueue( nextStatus2(currentStatus) )     //次の数を選択しない状態をキューに追加する
    endif
  endwhile
endfunction
```

図 2　関数 treeSearch のプログラム

〔探索回数の削減〕

関数 treeSearch で実装した方法では，n が大きくなるにつれて“取得した状態”を評価する回数（以下，探索回数という）も増大するが，不要な探索処理を行わないようにすることによって，③探索回数を削減することができる。探索回数の削減のために，探索を継続するかどうかを示すフラグを新たに用意し，次の (1)～(3) の処理を追加することにした。

(1)　“取得した状態”の合計が目標 X 以上の場合，以降の状態で数を選択しても合計は目標 X から離れてしまい，“解答の候補”にはならない。以降の状態の探索を不要とするために，フラグを探索中止に設定する。

(2)　(1) 以外の場合，フラグを探索継続に設定する。

(3)　フラグが探索中止の場合，“取得した状態”からの分岐を探索しないようにする。

探索回数の削減のために追加する変数を表 3 に示す。

表3　探索回数の削減のために追加する変数

名称	種類	内容
nextFlag	変数	“Y”のとき探索継続，“N”のとき探索中止を表す。

探索回数の削減を実装するために，図2中の（α）の行と（β）の行の間に図3のプログラムを追加し，（β）を“if（［　エ　］，かつ，nextFlagが“Y”である）”に修正した。

```
if( currentStatus.total が target 以上である )
  nextFlag ← [ オ ]
else
  nextFlag ← [ カ ]
endif
```

図3　探索回数の削減のための追加プログラム

設問1　図2中の［　ア　］～［　エ　］に入れる適切な字句を答えよ。

設問2　図3中の［　オ　］，［　カ　］に入れる適切な字句を答えよ。

設問3　本文中の下線①について，次の(1)，(2)の場合の評価の順序を，図1中の状態の記号（A）～（G）を用いてそれぞれ答えよ。ここで，分岐の際は左側のノードから先にデータ構造に格納することとする。本問では（D），（E），（F），（G）の後の状態は考慮しなくてよい。

(1)　〔探索の手順〕での記述どおり，データ構造にキューを使用した場合

(2)　本文中のキューを全てスタックに置き換えた場合

設問4　本文中の下線②について，データ構造にキューを使用した場合に，キューが必要とするメモリ使用量の最大値として適切な字句を解答群の中から選び，記号で答えよ。ここで，問題における数の個数をn，キューに状態を一つ格納するために必要なメモリ使用量をmとする。

解答群

ア　$2^n m$　　イ　$2nm$　　ウ　nm　　エ　$n^2 m$　　オ　$(n+1)m$

設問5　本文中の下線③における探索回数の削減を更に効率的に行うために，“指定されたn個の数”に実施しておくことが有効な事前処理の内容を20字以内で，その理由を25字以内でそれぞれ述べよ。