データ構造とプログラミング技法　　学科＿＿＿＿学籍番号＿＿＿＿＿＿氏名＿＿＿＿＿＿＿＿＿＿＿＿

1. 以下の文章の空欄を埋めなさい．(各 1 点（　　）は 1 組 1 点：合計 20 点)

- 関係 R が［　　律］（xRx），推移律（［　　］），［　　律］（xRyかつyRx ならば $x = y$）の 3 つを満足するとき関係 R を半順序関係と呼ぶ．これらの条件に加え，任意の x, y について（［　　］または［　　］）が常に成り立つとき，関係 R を全順序関係と呼ぶ．
- 全順序関係は［　　構造］，半順序関係は［　　構造］や，束構造における要素間の関係である．
- 物理構造としては，一連のデータを連続するメモリ番地に記憶する［　　］配置，メモリ上に分散したデータをポインタによって接続する［　　］配置の 2 種類がある．
- ヒープソートでは，キー列のデータ構造を，論理構造としては［　　］，物理構造としては［　　］配置と見なしている．
- 再帰呼び出しを用いた木構造の縦型探索アルゴリズムを，繰り返し計算を用いたアルゴリズムに変換する際には，［　　］という線形構造を用いる．また，木構造の横型探索アルゴリズムを実現するには［　　］という線形構造を用いる．
- ハッシュ法における衝突処理は，開番地法と［　　法］に分類される．開番地法の一つである［　　走査法］では，表サイズとハッシュ増分が互いに素である場合，探索周期は［　　］になる．
- キー列から逐次的に［　　探索木］を作成する際に，アンバランスな木構造になることがある．この問題を回避するために考案されたのが［　　木］である．この手法は木構造を作成している途中で発生する，平衡係数が［　］,［　］の範囲を超えた節のうち根から最も［い］ものについて，［　　回転］あるいは［　　回転］の操作を行って再平衡化を行うというものである．

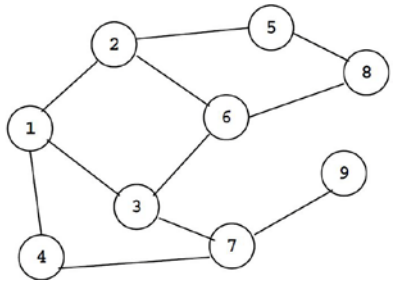
2. 下記はグラフ探索を行うプログラムである．空白部分を埋めなさい．（各 2 点：合計 6 点）

```
graph-search(vertex v)
{   [                ];
    while (!empty(S)) { pop(S,v);
                        if ( v.visit == 'NO' ) { v.visit = 'YES';
                                                 for (i=0; i<N;i++)
                                                 if ((adj[w[i]][v]==1)&&( [          ]== 'NO' )) [          ];
                        }
    }
}
```

但し，

- N は頂点数．w[i]は i 番目の頂点．adj[w][v]は隣接行列であり，頂点 w,v 間にエッジがあれば 1，なければ 0 の値がセットされている．
- 線形構造 S に対しては，vertex 型のデータ v を格納する push(S,v)と,v を取り出す pop(S,v)の操作のみが許されている．
- vertex 型のデータ v の visit 欄の初期値は'NO'であるとする．

3. 上記の関数のスタックをキューに変えて，下図のグラフの⑤を起点として探索を行った場合，節の番号を訪れる順番に並べなさい．但し，各節に書いてある番号 i はその頂点が i 番目であることを表す．また，push は enqueue，pop は dequeue に読み替えること．(6 点)



4. 下記のアルゴリズムの名称と目的を説明しなさい．また，空白部分を埋めなさい．(2+2+2=6 点)

```
void func1(int p)
{       int i,u,a;
        double tmp;
        add(p,S);
        for (i=0;i<N; i++) m[i]=L[p][i];
        while (remain()) {
                u = select_minimum();
                add(u,S);
                for (a=0; a<N; a++)
                        if (member(u,a)) { if (tmp= [          ]<m[a]) m[a]=tmp;}
 }
}
```

アルゴリズムの名称 ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

目的 ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿を求めるアルゴリズム．

5. 年功序列を基本とする給与体系を採用している会社の社員について，氏名，年齢，給与の 3 つの欄から成る名簿データがある．全社員に対するソーティングを 2 回行い，このデータを年齢の若い順に並べ，且つ，同じ年齢の場合は給与が少ない順に並ぶようにしたい．1，2 回目のソーティングに用いる欄はそれぞれ何か．また，2 回目のソーティングで使用すべきアルゴリズムとして何が適当かを理由とともに説明しなさい．(5 点)

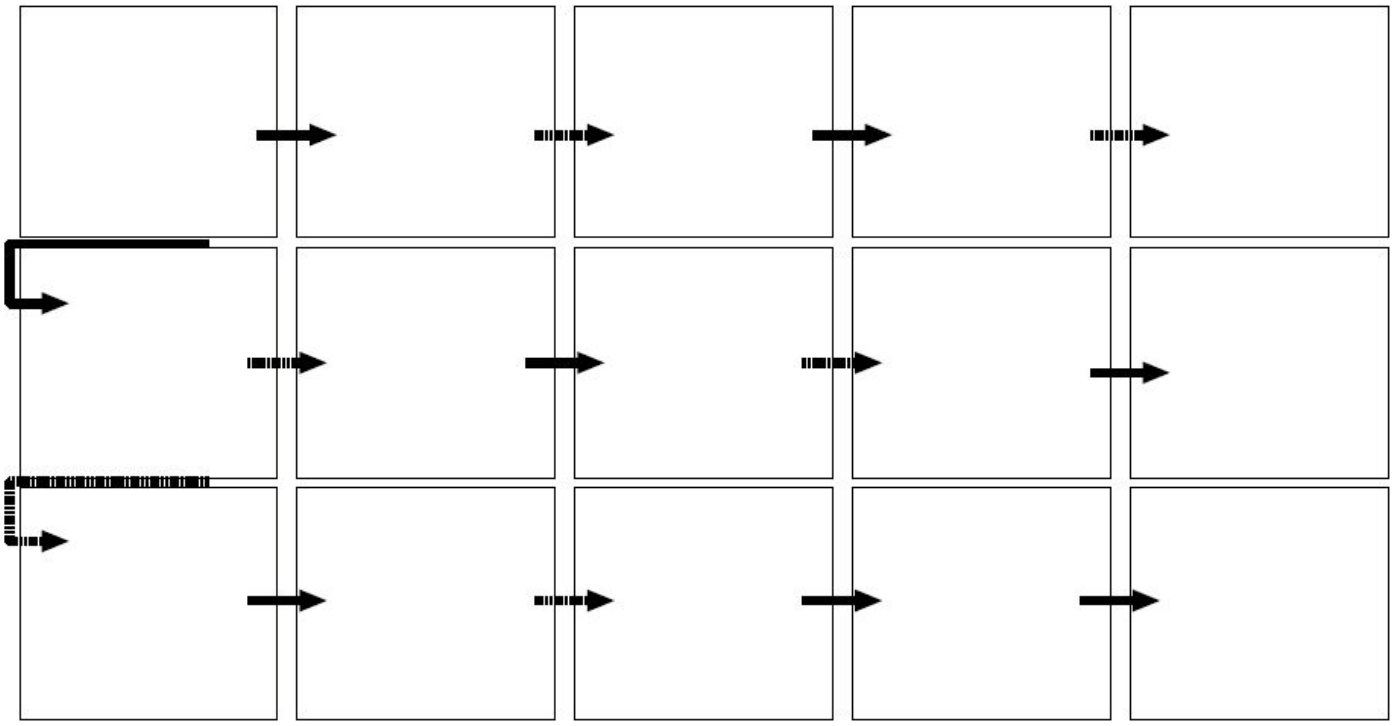
6. 2, 5,10,3,15, 19,14,11,8,6 を順に与えたときのハッシュ表と成功探索時の平均探査回数を求めなさい．但し，表サイズは 11，$h_0(x) = x \bmod 11, h_i(x) = (h_0(x) + 2 \times i) \bmod 11$ とする．(表 5 点，平均探査回数 3 点：合計 8 点)

表										
探査回数										

平均探査回数= 回

7. キーが 5, 15, 19, 1, 6, 11, 10, 9, 14 の順に配列 A[1]～A[9]に格納されているものとする．この配列が順配置された完全 2 分木である場合，その木を図示しなさい．また，これをヒープに変更した場合の木構造を示しなさい．(2 点＋4 点：合計 6 点)

8. 上の問題 7 で得られたヒープを用いてヒープソートを行う場合の木構造の変化を, 最初のヒープの段階から図示しなさい. (8 点)



9. 21, 2, 28, 14, 17, 5, 6 というデータを順に与えたときに出来上がる二分探索木と，平衡が崩れた際に再平衡化を行って作成した AVL 木を図示し, 成功探索時の平均探査回数をそれぞれ求めなさい. (2 分探索木 2 点, AVL 木 4 点，平均探査回数は各 2 点：合計 10 点)

10. 下記の KMP 法のプログラムにおいて, pat="BABAAB"の場合, compnext(pat)で計算している配列 next は，どのようになるか答えなさい．また，テキスト BABABBBABAABA から KMP 法で pat を見つける際の pat の配置と照合の様子を下図に書き込み，文字比較回数を答えなさい．(4 点+6 点：合計 10 点)

```
int kmp(char *text, char *pat)
{   int i=1, j=1;
    compnext(pat);
    while (j<=n) {
          while(i>0 && pat[i]!=text[j]) i=next[i];
          if ( i!=m ) {i++; j++;}
          else return j-m+1;
    }
    return -1;
}
```

next⇒

B	A	B	A	A	B

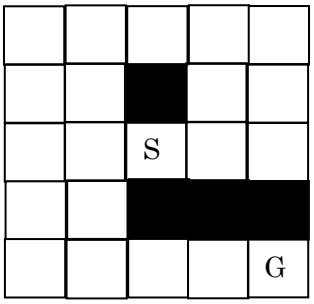
BABABBBABAABA（テキスト）

文字比較回数 ____回

11. 図の $\boldsymbol{S}=(x_S, y_S)$ からスタートして $\boldsymbol{G}=(x_G, y_G)$ に至る最短経路探索問題を考える．各マスにおいて移動可能な方向は，上下左右で，1 回の移動コストは，それぞれ 1 である．探索の過程で， $\boldsymbol{S}$ から $\boldsymbol{M}=(x_M, y_M)$ に到達している場合，そこから $\boldsymbol{G}$ に到る緩和解の評価関数を

$$g(\boldsymbol{M})=Cost(\boldsymbol{S},\boldsymbol{M})+|x_M-x_G|+|y_M-y_G|$$

とする．但し，$Cost(\boldsymbol{S},\boldsymbol{M})$ は $\boldsymbol{S} \Leftrightarrow \boldsymbol{M}$ の間の実移動コストの累積値である．このとき，best-first の縦型探索を用いて，分枝限定法で最短経路の探索を行う場合，探索が行われないマス（そのマスまでの実移動コストが計算されないマス）を塗りつぶし，最短経路の総移動コストを求めなさい．但し，黒いマスは障害物を表わし，コスト最小のノードは全て展開されると仮定する．(10 点)



12. 単純挿入法や線形探索などで用いられていた門番（番人，sentinel）は，アルゴリズムの上では用いても用いなくてもよい．それにもかかわらず，この技法が良く用いられる理由を説明しなさい．(5 点)

計算スペース----------

計算スペース

データ構造とプログラミング技法（解答例）　　学科＿＿＿＿学籍番号＿＿＿＿＿氏名＿＿＿＿＿

1. 以下の文章の空欄を埋めなさい．(各 1 点（ ＿＿ は 1 組 1 点）：合計 20 点)

- 関係 R が［ 反射律 ］（xRx），推移律（［ xRyかつyRz ならば xRz ］），［反対称律］（xRyかつyRx ならば $x=y$）の 3 つを満足するとき関係 R を半順序関係と呼ぶ．これらの条件に加え，任意の x, y について（［ xRy ］または［ yRx ］）が常に成り立つとき，関係 R を全順序関係と呼ぶ．
- 全順序関係は［ 線形 構造 ］，半順序関係は［ 木 構造 ］や，束構造における要素間の関係である．
- 物理構造としては，一連のデータを連続するメモリ番地に記憶する［ 順 ］配置，メモリ上に分散したデータをポインタによって接続する［ リンク ］配置の 2 種類がある．
- ヒープソートでは，キー列のデータ構造を，論理構造としては［ 完全二分木 ］，物理構造としては［ 順 ］配置と見なしている．
- 再帰呼び出しを用いた木構造の縦型探索アルゴリズムを，繰り返し計算を用いたアルゴリズムに変換する際には，［ スタック ］という線形構造を用いる．また，木構造の横型探索アルゴリズムを実現するには［キュー ］という線形構造を用いる．
- ハッシュ法における衝突処理は，開番地法と［ 連鎖　法］に分類される．開番地法の一つである［ 線形　走査法］では，表サイズとハッシュ増分が互いに素である場合，探索周期は［ 全表的 ］になる．
- キー列から逐次的に［ 二分探索木 ］を作成する際に，アンバランスな木構造になることがある．この問題を回避するために考案されたのが［ AVL 木］である．この手法は木構造を作成している途中で発生する，平衡係数が［-1, 1］の範囲を超えた節のうち根から最も［遠い］ものについて，［ 単 回転］あるいは［ 複 回転］の操作を行って再平衡化を行うというものである．

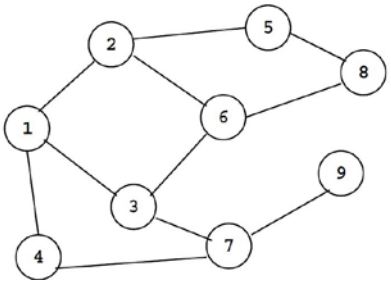
2. 下記はグラフ探索を行うプログラムである．空白部分を埋めなさい．（1 問 2 点：合計 6 点）

```
graph-search(vertex v)
{    push(S,v);
     while (!empty(S)) { pop(S,v);
                         if ( v.visit == 'NO' ) { v.visit = 'YES';
                                                  for (i=0; i<N;i++)
                                                  if ((adj[w[i]][v]==1)&&( w[i].visit== 'NO' )) push(S,w[i]);
                         }
     }
}
```

但し，

(ア) N は頂点数．w[i]は i 番目の頂点．adj[w][v]は隣接行列であり，頂点 w,v 間にエッジがあれば 1，なければ 0 の値がセットされている．
(イ) 線形構造 S に対しては，vertex 型のデータ v を格納する push(S,v)と,v を取り出す pop(S,v)の操作のみが許されている．
(ウ) vertex 型のデータ v の visit 欄の初期値は'NO'であるとする．

3. 上記の関数のスタックをキューに変えて，下図のグラフの⑤を起点として探索を行った場合，節の番号を訪れる順番に並べなさい．但し，各節に書いてある番号 i はその頂点が i 番目であることを表す．また，push は enqueue，pop は dequeue に読み替えること．(6 点)



5→2→8→1→6→3→7→4→9

4. 下記のアルゴリズムの名称と目的を説明しなさい．また，空白部分を埋めなさい．(2+2+2=6 点)

```
void func1(int p)
{       int i,u,a;
        double tmp;
        add(p,S);
        for (i=0;i<N; i++) m[i]=L[p][i];
        while (remain()) {
                u = select_minimum();
                add(u,S);
                for (a=0; a<N; a++)
                        if (member(u,a)) { if (tmp= m[u]+L[u][a]<m[a]) m[a]=tmp;}
 }
}
```

アルゴリズムの名称　Dijkstra のアルゴリズム

目的　（グラフ上の）頂点 p から各頂点までの距離（重みの和の最小値）を求める

5. 年功序列を基本とする給与体系を採用している会社の社員について，氏名，年齢，給与の 3 つの欄から成る名簿データがある．全社員に対するソーティングを 2 回行い，このデータを年齢の若い順に並べ，且つ，同じ年齢の場合は給与が少ない順に並ぶようにしたい．1，2 回目のソーティングに用いる欄はそれぞれ何か．また，2 回目のソーティングで使用すべきアルゴリズムとして何が適当かを理由とともに説明しなさい．(5 点)

まず 1 回目は，給与についてソーティングを行い，2 回目は年齢についてソーティングを行う.この場合,2 回目のソートは安定なソートでなければならない．安定なソートには単純挿入法，単純交換法，基数ソートなどがあるが，この場合には年功序列であるので，1 回目のソートで概ソート列が得られているため，単純交換法，単純挿入法などが適している．

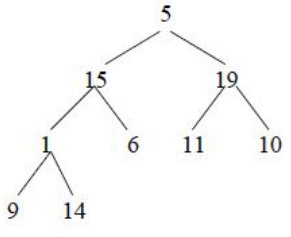
6. 2, 5,10,3,15, 19,14,11,8,6 を順に与えたときのハッシュ表と成功探索時の平均探査回数を求めなさい．但し，表サイズは 11，$h_0(x) = x \mod 11, h_i(x) = (h_0(x) + 2 \times i) \mod 11$ とする．(表 5 点，平均探査回数 3 点：合計 8 点)

表	11	8	2	3	15	5	6	14	19		10
探査回数	1	3	1	1	1	1	1	3	1		1

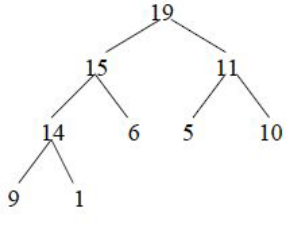
合計 14 回

平均探査回数=14/10=1.4 回

7. キーが 5, 15, 19, 1, 6, 11, 10, 9, 14 の順に配列 A[1]～A[9]に格納されているものとする．この配列が順配置された完全 2 分木である場合，その木を図示しなさい．また，これをヒープに変更した場合の木構造を示しなさい．(2 点＋4 点：合計 6 点)

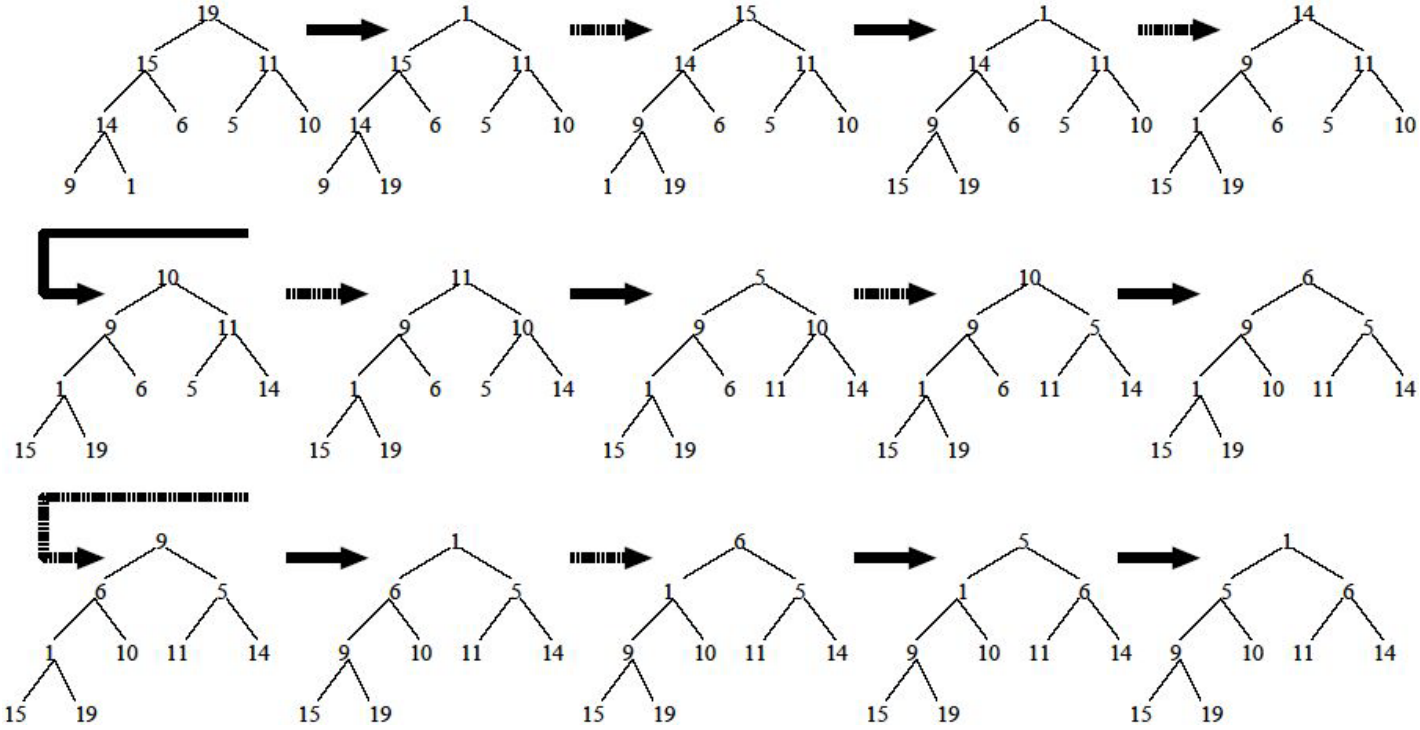


完全二分木

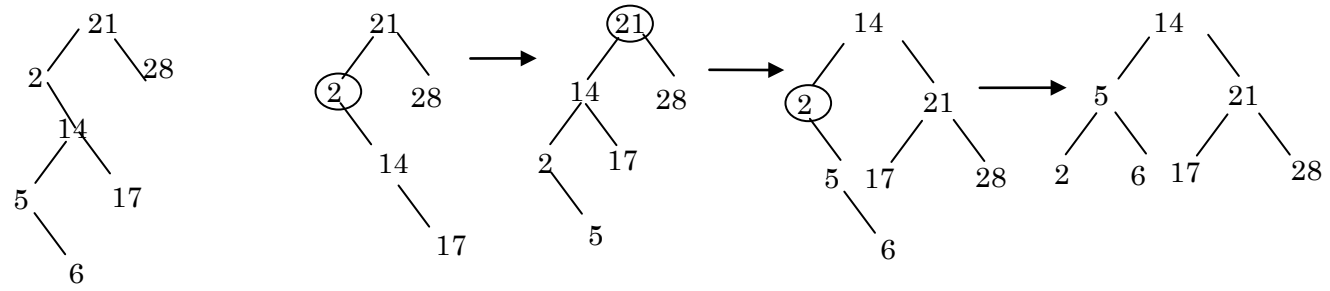


ヒープ

8. 上の問題 7 で得られたヒープを用いてヒープソートを行う場合の木構造の変化を, 最初のヒープの段階から図示しなさい. (8 点)



9. 21, 2, 28, 14, 17, 5, 6 というデータを順に与えたときに出来上がる二分探索木と, 平衡が崩れた際に再平衡化を行って作成した AVL 木を図示し, 成功探索時の平均探査回数をそれぞれ求めなさい. (2 分探索木 2 点, AVL 木 4 点, 平均探査回数は各 2 点 : 合計 10 点)



平均探査回数　(1+4+3+8+5)/7=3 回

平均探査回数　17/7=2.43 回

10. 下記の KMP 法のプログラムにおいて, pat="BABAAB"の場合, compnext(pat)で計算している配列 next は, どのようになるか答えなさい. また, テキスト BABABBBABAABA から KMP 法で pat を見つける際の pat の配置と照合の様子を下図に書き込み, 文字比較回数を答えなさい. (4 点+6 点 : 合計 10 点)

```
int kmp(char *text, char *pat)
{   int i=1, j=1;
    compnext(pat);
    while (j<=n) {
        while(i>0 && pat[i]!=text[j]) i=next[i];
        if ( i!=m ) {i++; j++;}
        else return j-m+1;
    }
    return -1;
}
```

next⇒

B	A	B	A	A	B
0	1	0	1	3	0

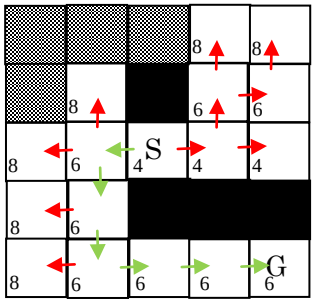
```
BABABBBABAABA (テキスト)
BABAA
   BABA
      BA
         BABAAB
文字比較回数　15 回
```

11. 図の $\boldsymbol{S}=(x_S, y_S)$ からスタートして $\boldsymbol{G}=(x_G, y_G)$ に至る最短経路探索問題を考える．各マスにおいて移動可能な方向は，上下左右で，1 回の移動コストは，それぞれ 1 である．探索の過程で， $\boldsymbol{S}$ から $\boldsymbol{M}=(x_M, y_M)$ に到達している場合，そこから $\boldsymbol{G}$ に到る緩和解の評価関数を

$$g(\boldsymbol{M})=Cost(\boldsymbol{S},\boldsymbol{M})+|x_M-x_G|+|y_M-y_G|$$

とする．但し，$Cost(\boldsymbol{S},\boldsymbol{M})$ は $\boldsymbol{S} \Leftrightarrow \boldsymbol{M}$ の間の実移動コストの累積値である．このとき，best-first の縦型探索を用いて，分枝限定法で最短経路の探索を行う場合，探索が行われないマス（そのマスまでの実移動コストが計算されないマス）を塗りつぶし，最短経路の総移動コストを求めなさい．但し，黒いマスは障害物を表わし，コスト最小のノードは全て展開されると仮定する．(10 点)

右図のように探索が行われる．（最初は S から見て 2 方向に進み，best first サーチなので $g(\boldsymbol{M})=4$ となる右向き矢印の方向が選ばれる．しかし，障害物のため $g(\boldsymbol{M})=4$ のパスはゴールに到達しないことが分かり， $g(\boldsymbol{M})=6$ のパスが評価される． $g(\boldsymbol{M})=6$ のノードは，S から見て，右斜め上と，左に存在するが，右斜め上の $g(\boldsymbol{M})=6$ のパスはゴールにたどり着かない．このため，S から見て，左に伸びている $g(\boldsymbol{M})=6$ のパスが G につながっており，暫定値は $z=6$ となる．既に展開された各ノードの $g(\boldsymbol{M})$ は最小でも 8 であり，全て z の値よりも大きいので，全て終端されてしまい，これ以上探索は行われない．したがって，灰色のマスが探索されない．総移動コストは 6 である．



12. 単純挿入法や線形探索などで用いられていた門番（番人，sentinel）は，アルゴリズムの上では用いても用いなくてもよい．それにもかかわらず，この技法が良く用いられる理由を説明しなさい．(5 点)

門番を用いると，ループ内部の条件分岐命令が一つ少なくなる．この命令が減ることにより，1 命令が減る以上に速度が向上する．これは，CPU 内部の命令パイプラインの実行が条件分岐命令の実行によって滞るためである．このパイプラインの乱れは，条件が確定しない限り次の命令が格納されている番地が確定できないという条件分岐命令の性質に依るものであり，この命令を減らせば確実に実行速度が向上する．最近の CPU には，条件部の計算が終了していない場合にも，実行される可能性のある命令を全て実行しておいて，条件部の計算が済んでから，実行されない命令の実行結果を捨て去る「投機的実行」機構が組み込まれたものもあるが，それでも条件分岐が重なると実行速度が低下するため，門番の使用は効果的である．