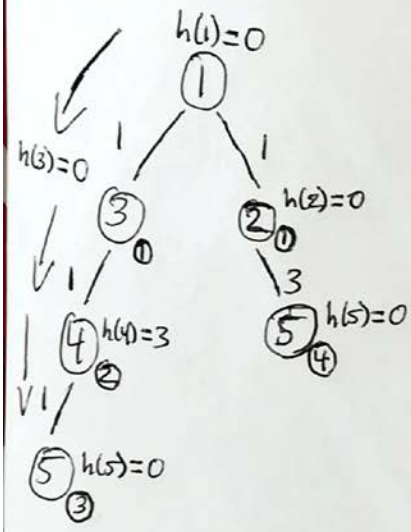
1a)

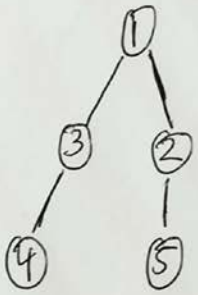
1a) Uniform search algorithm:



Visited: 1, 2, 3, 4

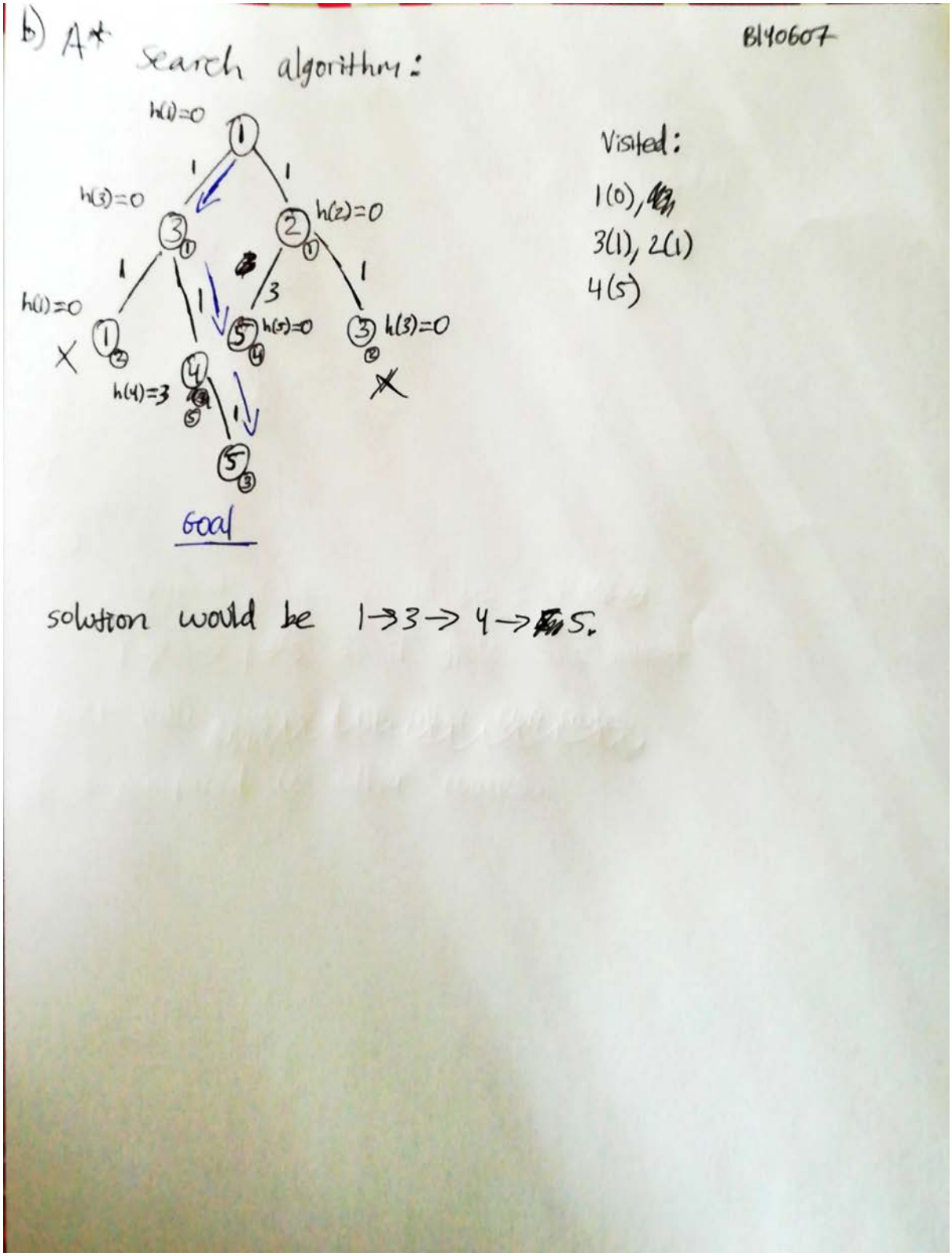
The cheapest route would be : 1→3→4→5 as it yields the lowest cost value of 3 compared to other routes.

graph search: Explored: 1, 3, 2, 4, 5



Solution would be [1, 3, 2, 4, 5]

b)


b) A* search algorithm:
B140607
h(1)=0
h(3)=0
h(2)=0
h(1)=0
h(5)=0
h(3)=0
h(4)=3
Goal
Visited:
1(0),
3(1), 2(1)
4(5)
solution would be 1→3→4→5.

c) Uniform-cost search would be optimal because, at every step the path with the least cost is chosen, and paths never gets shorter as nodes are added, ensuring that the search expands nodes in the order of their optimal path cost. A* search would also be optimal if the heuristic is admissible meaning that

d)

e) No, the search strategies would not be optimal if TREE-SEARCH is used because in the case of a tree search, we do not keep a closed list. Consequently, the same node can be visited multiple (or even infinitely many) times, which means that the produced tree (by the tree search) may contain the same node multiple times.

2a) The syntax of FOL is that it consists of:

Constants which are named objects e.g. KingJohn, 2, UoE, ...

Predicates (Relations) e.g. Prime, >, HotterThan, ...

Functions such as Sqrt, LeftLegOf, ...

Variables such as x, y, a, b. By convention, variables start with lower-case letters (lest they might be confused with constants.)

Connectives such as ¬,⟹,∧,∨,⟺, …

Equality *e.g.* ==. Equality compares whether two objects are equal, regardless of their name.

Quantifiers such as ∀,∃,∃!, …

Sentences such as Brother(Richard, John)

2b) FOL brings structures to facts which can be built from:

Objects such as people, cars

Functions like father of

Relations such as bigger than

A first order signature is a pair (F, P) where,

F - indexed family (F_n) n ∈ $\mathbb{N}$ of sets of function symbols (operations)

P – indexed family (P_n) n ∈ $\mathbb{N}$ of sets of relation symbols (predicates)

For σ ∈ F_n and π ∈ P_n, n is called arity.

Constant symbols are function symbols with arity zero.

c)

e) Most general unifier is a unifier σ of the terms such that any other unifier can be written as σ∘τσ∘τ for some substitution ττ.

f) The unification algorithm would be:

```
global σ : substitution;   { Initialized to Id }

Unify( s : term; t : term )
  begin
    if s is a variable then   { Instantiate variables }
       s := sσ;
    if t is a variable then
       t := tσ;
    if s is a variable and s = t then
       { Do nothing }
    else if s = f(s_1,....s_n) and t = g(t_1,....,t_m) for n,m ≥ 0 then begin
       if f = g then
          for i := 1 to n do
             Unify( s_i, t_i );
       else Exit with failure   { Symbol clash }
    end
    else if s is not a variable then
       Unify( t, s );
    else if s occurs in t then
       Exit with failure;   { Occurs check }
    else σ := σ{s → t};
  end;
```