figs

Mahmudur Rahman

May 2018

1 Algorithm

Here we will write algorithms. For starter, let us write an algorithm that is popularly known as the Dijkstra's algorithm.

Algorithm 1: How to write algorithms

Result: Finds the factorial of a number

```
1  x ← value;
2  while While condition do
3  |  instructions;
4  |  if x ≤ y then
5  |  |  instructions1;
6  |  |  instructions2;
7  |  end
8  |  else if x ≤ y then
9  |  |  instructions1;
10 |  |  instructions2;
11 |  end
12 |  else
13 |  |  instructions3;
14 |  end
15 end
```

Okay. Let us look at another.

2 Code

```
#include<iostream>

using namespace std;

class Point2D
{
    double x,y;
public:
    Point2D(){ cout << "Point2D def con\n"; x = 0; y = 0; } //
    default constructor initializes to (0,0)
```

```
10      Point2D(double x, double y);
11      void setX(double x);
12      void setY(double y);
13      double getX();
14      double getY();
15      void print();
16
17      Point2D operator++();
18      Point2D operator+(Point2D P2);
19      Point2D operator*(double n);
20      bool operator==(Point2D p1);
21      bool operator!=(Point2D p1);
22
23      ~Point2D(){ cout << "Point2D dest\n"; x = 0; y = 0; } //
        destructor that sets all values to 0
24  };
25
26  Point2D Point2D::operator++()
27  {
28      x++;
29      y++;
30      return (*this);
31  }
32
33
34  Point2D::Point2D(double argx,double argy)
35  {
36      cout << "Point2D 2 param con\n";
37      x = argx;
38      y = argy;
39  }
40
41
42  Point2D Point2D::operator+(Point2D P2)
43  {
44      Point2D P;
45      P.x = P2.x + x;
46      P.y = P2.y + y;
47      return P;
48  }
49
50  Point2D Point2D::operator*(double n)
51  {
52      Point2D P;
53      P.x = x * n;
54      P.y = y * n;
55      return P;
56  }
57
58  bool Point2D::operator==(Point2D P1)
59  {
60      if(x == P1.x && y == P1.y){
61          return true;
62      }
63      return false;
64  }
65
```

```
66  bool Point2D::operator!=(Point2D P1)
67  {
68      if(x != P1.x || y != P1.y){
69          return true;
70      }
71      return false;
72  }
73
74
75
76  void Point2D::setX(double argx)
77  {
78      //Complete this function
79      x = argx;
80  }
81
82  void Point2D::setY(double argy)
83  {
84      y = argy;
85  }
86
87  double Point2D::getX()
88  {
89      return x;
90  }
91
92  double Point2D::getY()
93  {
94      //Complete this function
95      return y;
96  }
97
98  void Point2D::print()
99  {
100     cout << "(" << x << "," << y << ")";
101 }
102
103 class Point3D : public Point2D
104 {
105     double z;
106 public:
107     Point3D();
108     Point3D(double argx, double argy, double argz);
109     void setZ(double argz) { z = argz; }
110     double getZ() { return z; }
111     void print();
112     Point3D operator++();
113     ~Point3D() { cout << "Point3D dest\n"; z = 0; }
114     bool operator==(Point3D rhs);
115 };
116
117 Point3D::Point3D()
118 {
119     cout << "Point3D def con";
120     z = 0;
121 }
122
```

```
123 Point3D::Point3D(double argx, double argy, double argz)
124     :Point2D(argx,argy)
125 {
126     cout << "Point3D 3 param con";
127     z = argz;
128 }
129
130 Point3D Point3D::operator++()
131 {
132     Point2D::operator++();
133     z++;
134     return (*this);
135 }
136
137 bool Point3D::operator==(Point3D rhs)
138 {
139     if( Point2D::operator==(rhs) && z==rhs.z)
140         return true;
141     else return false;
142 }
143
144 void Point3D::print()
145 {
146     cout << "(" << getX() << "," << getY() << "," << z << ")";
147 }
148
149
150 int main(void)
151 {
152     Point3D p1(10,20,30);
153     Point3D p2(10,20,30);
154     if(p1==p2) cout << "Equal\n";
155     else cout << "Not equal\n";
156     ++p1;
157     p1.print();
158     cout << endl;
159     return 0;
160 }
```

3 Figures

4 Introduction

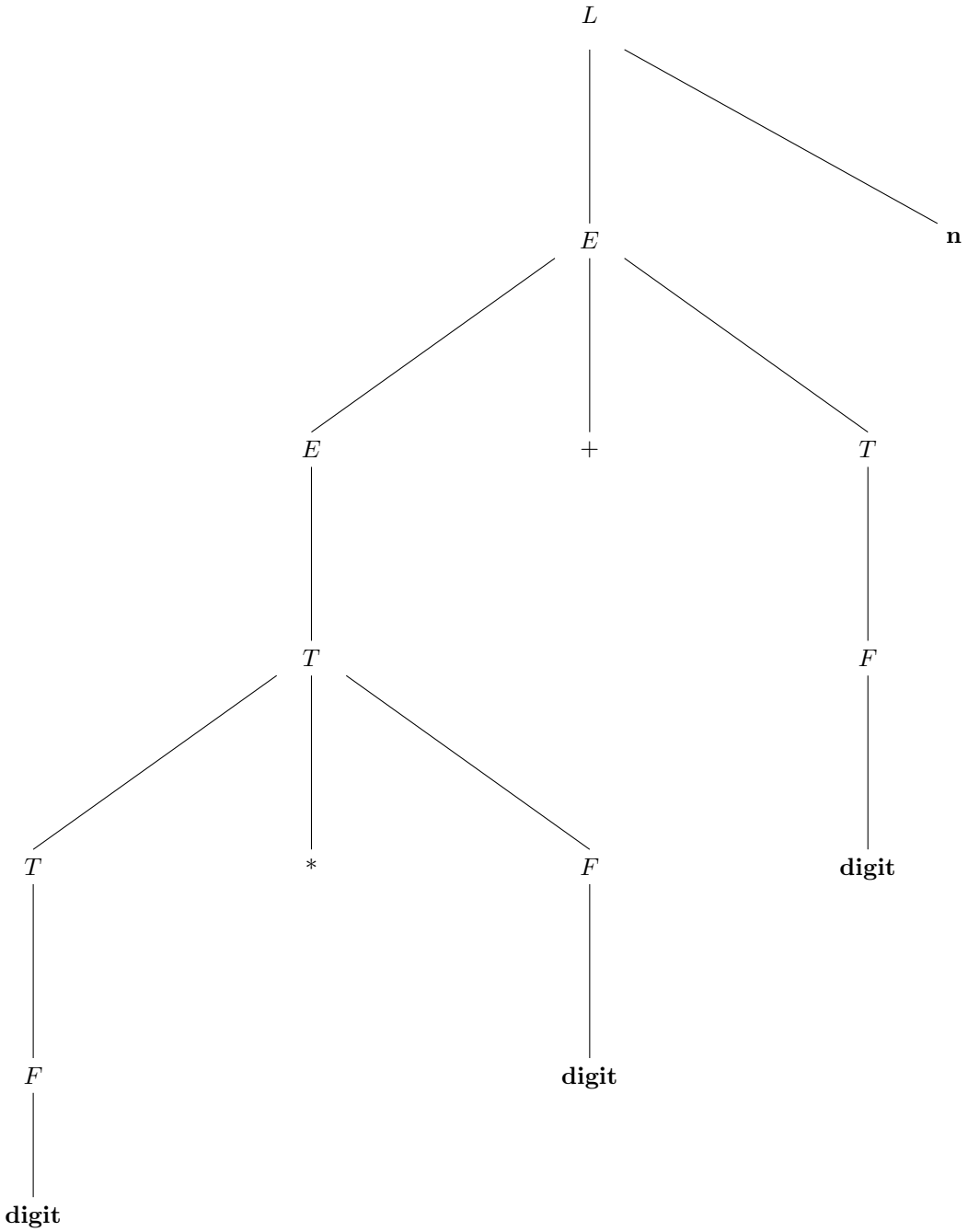

Figure 1: Vector image

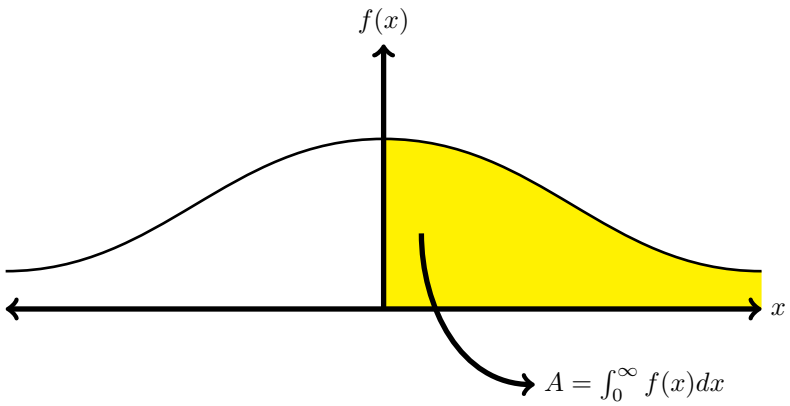

Figure 2: Test

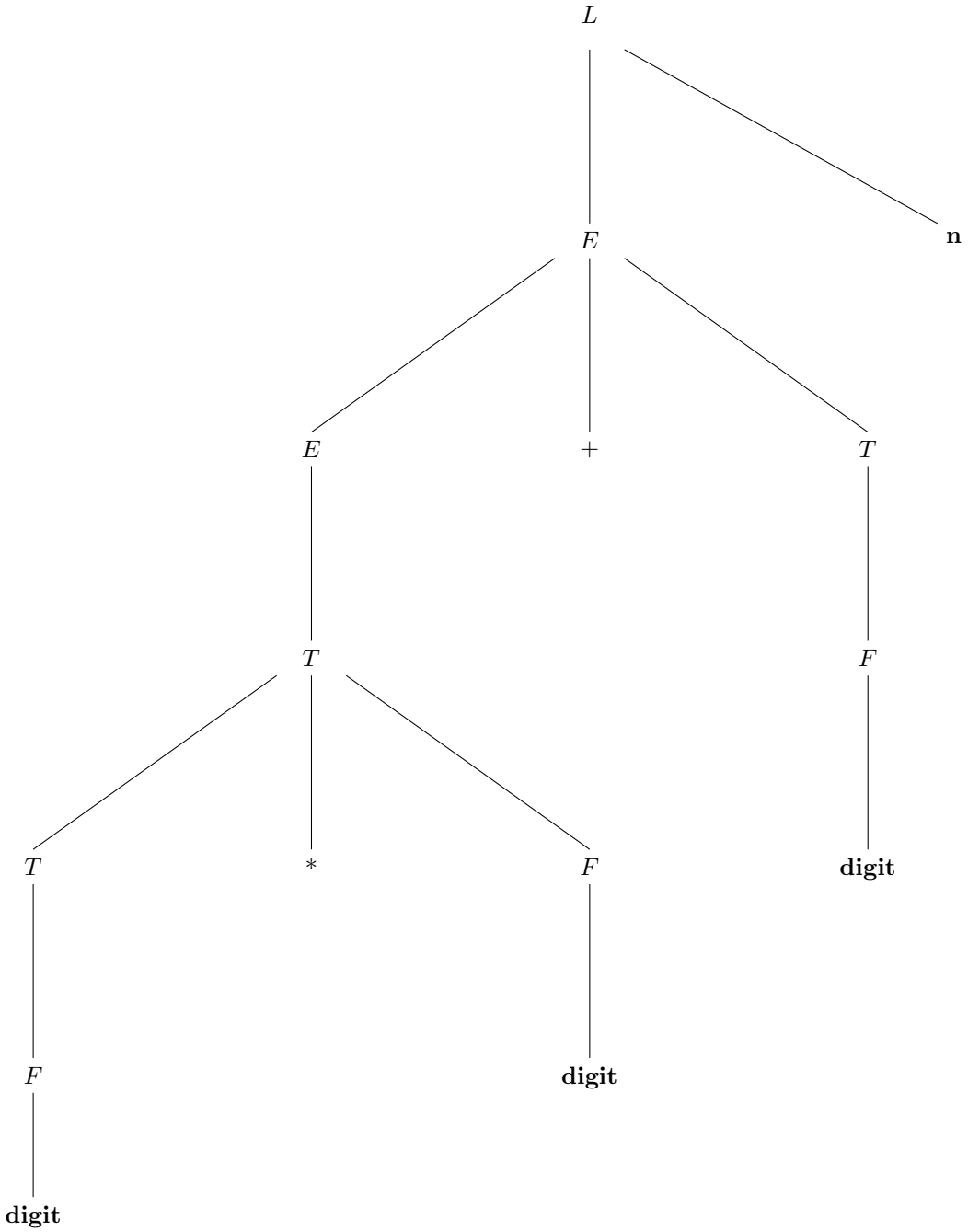

Figure 3: Vector image

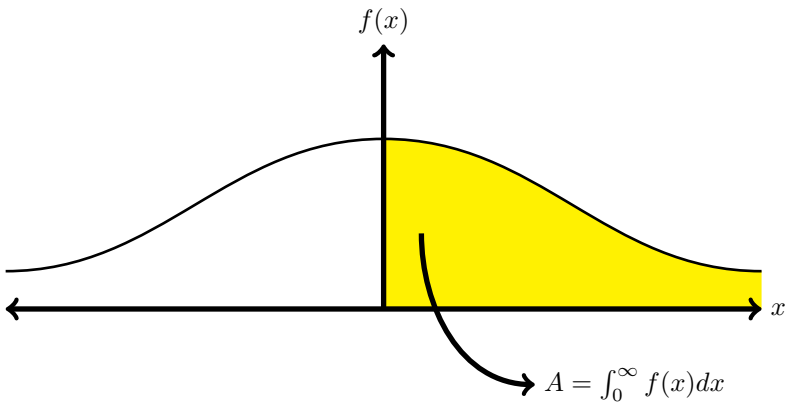

Figure 4: Test