figs

Mahmudur Rahman

May 2018

1 Figure

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
$x \leftarrow value$;
while While condition do
    instructions;
    if $x \leq y$ then
        instructions1;
        instructions2;
    end
    else if $x \leq y$ then
        instructions1;
        instructions2;
    end
    else
        instructions3;
    end
end
```

Okay. Let us look at another.

```
#include<iostream>

using namespace std;

class Point2D
{
    double x,y;
public:
    Point2D(){ cout << ”Point2D␣def␣con\n”; x = 0; y = 0; } //default constructo
    Point2D(double x, double y);
    void setX(double x);
```

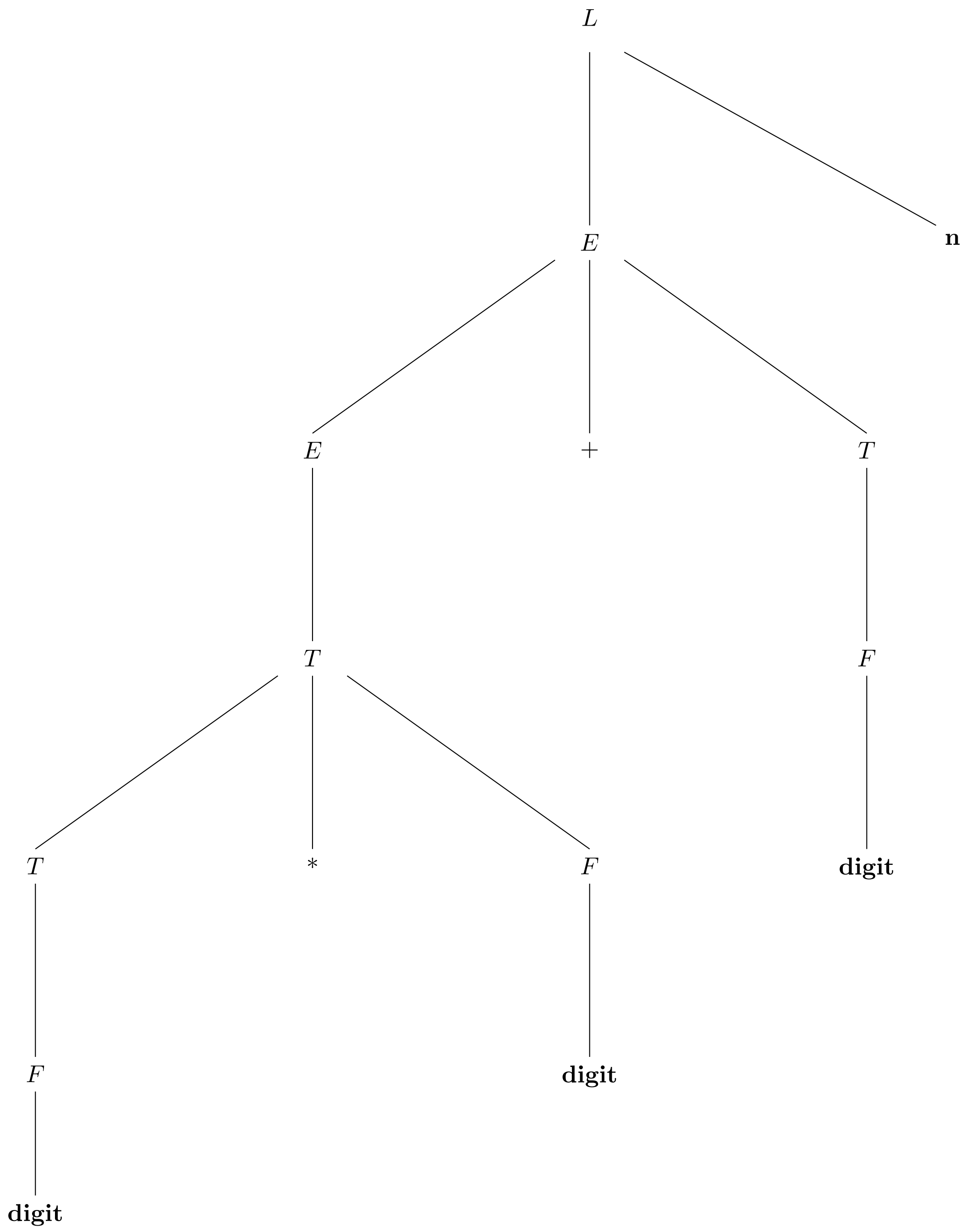

Figure 1: Vector image

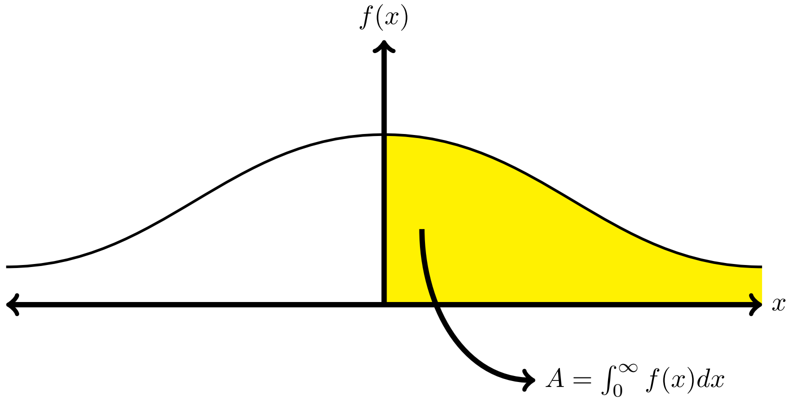

Figure 2: Test

```
    void setY(double y);
    double getX();
    double getY();
    void print();

    Point2D operator++();
    Point2D operator+(Point2D P2);
    Point2D operator*(double n);
    bool operator==(Point2D p1);
    bool operator!=(Point2D p1);

    ~Point2D(){ cout << "Point2D␣dest\n"; x = 0; y = 0; } //destructor that sets
};

Point2D Point2D::operator++()
{
    x++;
    y++;
    return (*this);
}


Point2D::Point2D(double argx,double argy)
{
    cout << "Point2D␣2␣param␣con\n";
    x = argx;
    y = argy;
}
```

```
Point2D Point2D::operator+(Point2D P2)
{
    Point2D P;
    P.x = P2.x + x;
    P.y = P2.y + y;
    return P;
}

Point2D Point2D::operator*(double n)
{
    Point2D P;
    P.x = x * n;
    P.y = y * n;
    return P;
}

bool Point2D::operator==(Point2D P1)
{
    if(x == P1.x && y == P1.y){
        return true;
    }
    return false;
}

bool Point2D::operator!=(Point2D P1)
{
    if(x != P1.x || y != P1.y){
        return true;
    }
    return false;
}



void Point2D::setX(double argx)
{
    //Complete this function
    x = argx;
}

void Point2D::setY(double argy)
{
    y = argy;
}
```

```
double Point2D::getX()
{
    return x;
}

double Point2D::getY()
{
    //Complete this function
    return y;
}

void Point2D::print()
{
    cout << "(" << x << "," << y << ")";
}

class Point3D : public Point2D
{
    double z;
public:
    Point3D();
    Point3D(double argx, double argy, double argz);
    void setZ(double argz) { z = argz; }
    double getZ() { return z; }
    void print();
    Point3D operator++();
    ~Point3D() { cout << "Point3D␣dest\n"; z = 0; }
    bool operator==(Point3D rhs);
};

Point3D::Point3D()
{
    cout << "Point3D␣def␣con";
    z = 0;
}

Point3D::Point3D(double argx, double argy, double argz)
    :Point2D(argx,argy)
{
    cout << "Point3D␣3␣param␣con";
    z = argz;
}

Point3D Point3D::operator++()
{
    Point2D::operator++();
    z++;
    return (*this);
}

bool Point3D::operator==(Point3D rhs)
{
    if( Point2D::operator==(rhs) && z==rhs.z)
        return true;
    else return false;
}

void Point3D::print()
{
    cout << "(" << getX() << "," << getY() << "," << z << ")";
}


int main(void)
{
    Point3D p1(10,20,30);
    Point3D p2(10,20,30);
    if(p1==p2) cout << "Equal\n";
    else cout << "Not␣equal\n";
    ++p1;
    p1.print();
    cout << endl;
    return 0;
}
```