



Assembly Line Scheduling

*Supervised by Dr. Issam
Salman*



الجامعة الافتراضية السورية
SYRIAN VIRTUAL UNIVERSITY

Intelligent Algorithms

Assembly Line Scheduling

Program Name : ITE
Semester : S24
Course Code : BIA601
Doctor : Issam Salman

The participating students

Full name	Username	My class
عطاف عثمان	Etaf_154254	C1
أسامة جمعة	Osama_177186	C2
قمر سيلان	Kamar_180726	C2
ماجدة عبد الله الجندي	Majedah_137770	C3
راما الهندي	Rama_158075	C2
عمر الجنيد	Omar_154129	C2
حسن زيدان	Hasan_171117	C3

GitHub repository link:

https://github.com/osamaju/ITE_BID601_HW.git

مقدمة

جدولة خطوط التجميع تُعد واحدة من أهم التحديات التي تواجهها الصناعات الحديثة في مجال التصنيع وإدارة العمليات. تعتمد هذه العملية على تنظيم وتنسيق سلسلة من المهام التي يتم تنفيذها عبر محطات عمل متتالية في خط التجميع، بهدف تحويل المواد الخام إلى منتج نهائي بأعلى كفاءة ممكنة.

الهدف الأساسي من جدولة خطوط التجميع هو تقليل الوقت الإجمالي للإنتاج، المعروف باسم "زمن الإنجاز"، مع الحفاظ على جودة المنتج ومراعاة القيود الفنية واللوجستية المرتبطة بعملية التصنيع.

في خط التجميع، يمر المنتج عبر عدة مراحل، حيث تقوم كل محطة عمل بمهمة محددة، مثل التجميع، الفحص، أو التغليف. التحدي الرئيسي هنا هو توزيع المهام على المحطات بشكل متوازن، بحيث لا تتراكم المهام في محطة واحدة بينما تكون محطات أخرى خاملة. هذا التوازن ضروري لضمان استمرارية العمل دون تأخير أو تكاليف إضافية. تزداد تعقيدات هذه العملية عند وجود خطوط تجميع متعددة تعمل بالتوازي، أو عندما تكون هناك تبعيات زمنية بين المهام، حيث يتطلب بعضها الانتهاء قبل بدء مهام أخرى.

تكتسب جدولة خطوط التجميع أهمية كبيرة في الصناعات المختلفة بسبب تأثيرها المباشر على الإنتاجية وتكاليف التشغيل. على سبيل المثال، في صناعة السيارات، يمكن أن تؤدي الجدولة الفعالة إلى تقليل وقت التوقف عن العمل وزيادة عدد السيارات المنتجة في نفس الفترة الزمنية. وفي صناعة الإلكترونيات، تساعد الجدولة الدقيقة في تلبية الطلبات المتزايدة على الأجهزة الإلكترونية بسرعة وكفاءة.

بالإضافة إلى ذلك، تعتمد العديد من الشركات على تقنيات متقدمة مثل الذكاء الاصطناعي والأتمتة لتحسين عمليات الجدولة. هذه التقنيات تسمح بتحليل كميات كبيرة من البيانات وتحديد أفضل الطرق لتوزيع المهام، مما يؤدي إلى تحسين استخدام الموارد وتقليل الهدر.

تعتبر جدولة خطوط التجميع عنصرًا حيويًا في نجاح أي عملية تصنيعية، حيث تساهم في تحقيق أهداف الشركة من حيث خفض التكاليف، زيادة الإنتاجية، وتحسين جودة المنتجات. مع تطور التكنولوجيا وزيادة تعقيد العمليات الصناعية، تظل الجدولة الفعالة لخطوط التجميع مجالًا خصبًا للبحث والابتكار.

Assembly Line Scheduling	1
مقدمة	4
Task Allocation in Teams	7
Frontend	7
Backend.....	7
Preparing Hosting.....	7
Report Formatting	7
Web Interfaces(Front-end)	8
Index	8
Input.....	9
Dynamic Fields Display	11
Entering Valid Values	14
Result	16
Back end	17
Programming Language : Python.....	17
Framework : Flask	17
Hosting : PythonAnywhere.....	17
Project Structure.....	18
app.py	18
<i>Flask</i> استيراد المكتبات وإنشاء تطبيق	19
دالة <i>assembly_line_scheduling</i>	19
حساب أوقات الدخول الأولية	20
حساب الوقت الأمثل لكل محطة	20
حساب الوقت الأمثل النهائي	20
بناء المسار الأمثل	20
مسار الصفحة الرئيسية	21
مسار صفحة الإدخال	21
دالة <i>input_page</i>	21
التحقق من طريقة الطلب (<i>Request Method</i>).....	21
، أي أن المستخدم قام بإرسال البيانات. <i>POST</i> يتم التحقق من أن الطلب المرسل من النموذج هو من نوع	21
:تم استخراج البيانات المدخلة من النموذج	22
عدد المحطات : <i>n</i>	22
أوقات الدخول للخطين 1 و 2 : <i>e1, e2</i>	22
أوقات الخروج للخطين 1 و 2 : <i>x1, x2</i>	22
<i>int()</i> يتم تحويل هذه البيانات إلى أعداد صحيحة باستخدام	22
تعبئة مصفوفات أوقات المحطات	22

assembly_line_scheduling.....	23
استدعاء دالة	
عرض النتائج :	23
عرض صفحة الإدخال (إذا لم يتم إرسال النموذج)	23
تشغيل التطبيق	24
templates/	24
static/	24
وصف الخوارزمية Assembly Line Scheduling	28

Task Allocation in Teams

Frontend

من ناحية تطوير واجهات الويب وتجهيزها قام بذلك **أسامة و قمر**

Backend

تجهيز خوارزمية خطوط الإنتاج و ربطها بالموقع قام بذلك **راما و حسن**

Preparing Hosting

عمر قام باستضافة الموقع على pythonanywhere حيث اعتمد في الباك ايند بايثون و فلاسك

Report Formatting

تم تنسيق التقرير و تهيئته من قبل **ماجدة و عطا**

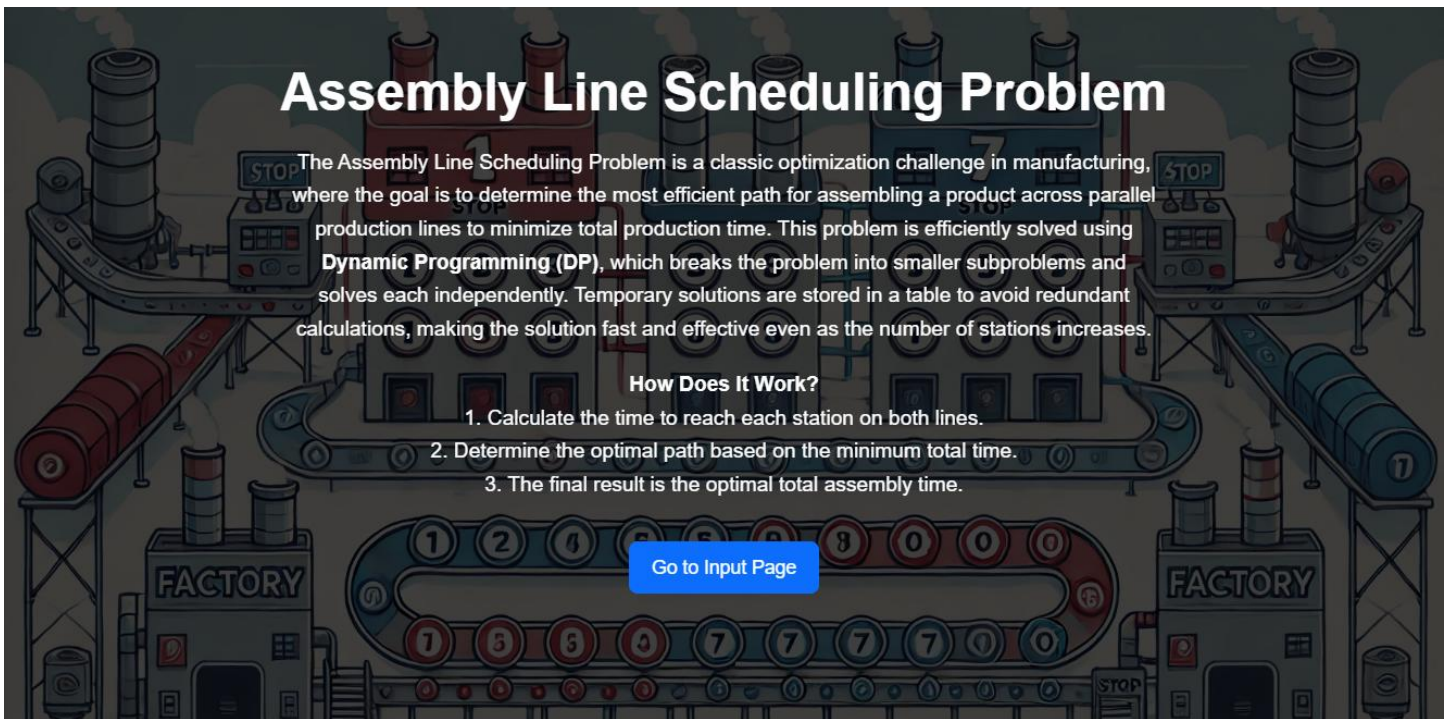
Web Interfaces(Front-end)

يستطيع المستخدم الدخول للموقع عن طريق الرابط التالي :

[/https://osama123123.pythonanywhere.com](https://osama123123.pythonanywhere.com)

حيث تمت استضافة الموقع على موقع pythonanywhere

Index



الصفحة الرئيسية في الموقع تحتوي وصفاً عاماً للمشكلة التي يقوم موقعنا بحلها و كيفية عملها بشكل عام و زر للذهاب لصفحة الادخال

Input

Assembly Line Scheduling

Number of Stations (n):

Enter a number greater than 0.

Entry Time for Line 1 (e1): Entry Time for Line 2 (e2):

Exit Time for Line 1 (x1): Exit Time for Line 2 (x2):

صفحة ادخال البيانات حيث يتوجب على المستخدم ادخال عدد المحطات اولاً
عند ادخال قيمة اكبر من 0 سيتم انشاء حقول لإدخال تكلفة كل محطة بالإضافة لأزمنة الانتقال
بين المحطات متناسبة مع عدد المحطات التي ادخلها المستخدم وتم القيام بذلك عبر الجافا سكربت

```
// Function to dynamically update the form fields based on the number of stations (n)
function updateFields () {
  const n = parseInt(document.getElementById('n').value)
  const aFields = document.getElementById('a-fields')
  const tFields = document.getElementById('t-fields')

  // Clear previous fields
  aFields.innerHTML = ''
  tFields.innerHTML = ''

  if (n > 0) {
    // Add station times (a) fields
    aFields.innerHTML += '<h2>Station Times (a):</h2>'
    for (let i = 0; i < 2; i++) {
      aFields.innerHTML += `<h3>Line ${i + 1}:</h3>`
      let row = '<div class="input-row">'
      for (let j = 0; j < n; j++) {
        row += `
          <div class="form-group">
            <input type="text" value="0">
          </div>
        `
      }
      row += '</div>'
      aFields.innerHTML += row
    }
  }
}
```

```

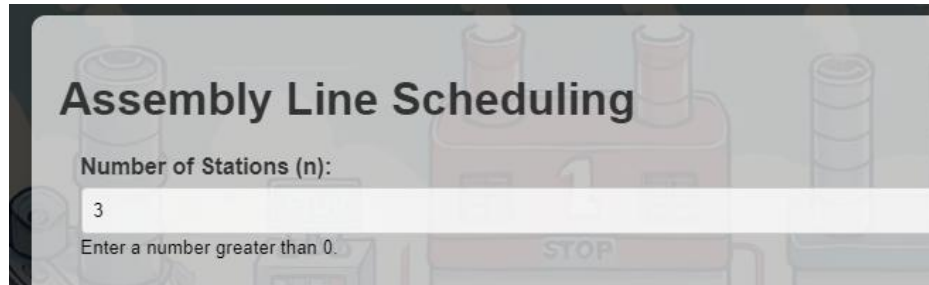
        <label for="a${i}${j}">a${i + 1}${j + 1}:</label>
        <input type="number" id="a${i}${j}" name="a${i}${j}"
required min="0">
        </div>`
    }
    row += '</div>'
    aFields.innerHTML += row
}

// Add transfer times (t) fields
tFields.innerHTML += '<h2>Transfer Times (t):</h2>'
for (let i = 0; i < 2; i++) {
    tFields.innerHTML += `<h3>Line ${i + 1}:</h3>`
    let row = '<div class="input-row">'
    for (let j = 0; j < n - 1; j++) {
        row += `
            <div class="form-group">
                <label for="t${i}${j}">t${i + 1}${j + 1}:</label>
                <input type="number" id="t${i}${j}" name="t${i}${j}"
required min="0">
            </div>`
        }
        row += '</div>'
        tFields.innerHTML += row
    }
}
}
}

```

نقوم بأخذ قيمة عدد المحطات من العنصر ذو ال id "N" وانشاء حقول الادخال بناء على القيمة المدخلة من قبل المستخدم واطافة الحقول ك html الى الحقلين 'a-fields' و 't-fields'

Dynamic Fields Display



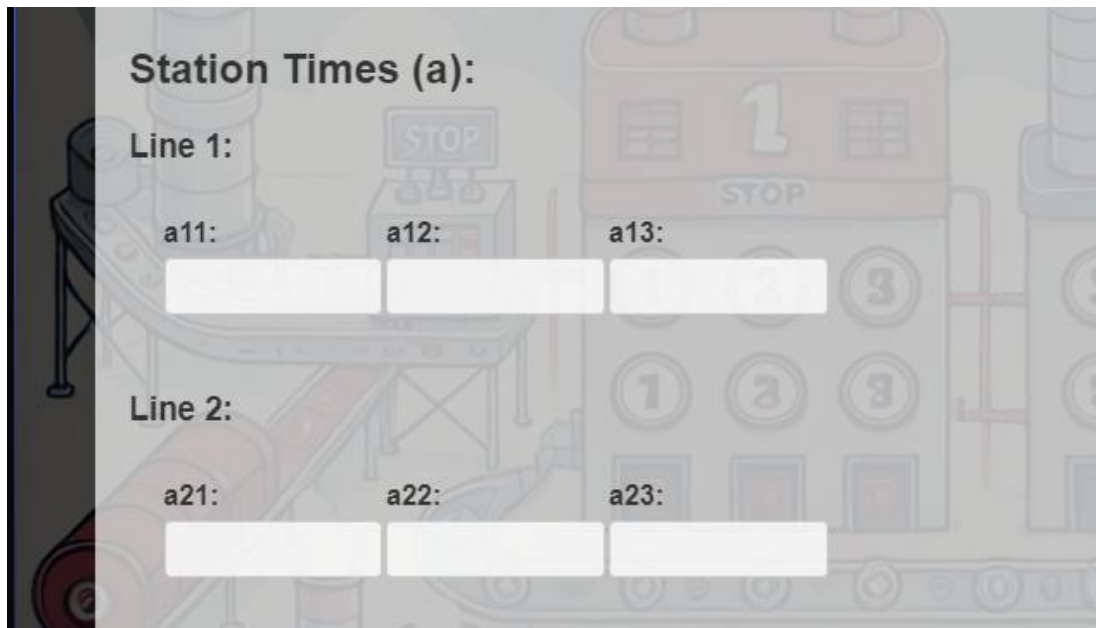
Assembly Line Scheduling

Number of Stations (n):

3

Enter a number greater than 0.

عند ادخال رقم 3 سنجد ظهور ثلاث حقول ادخال



Station Times (a):

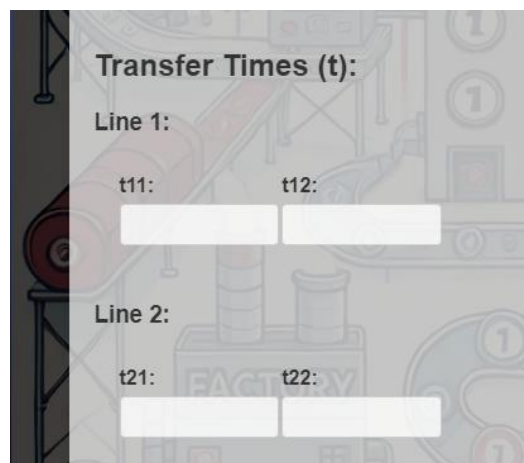
Line 1:

a11: a12: a13:

Line 2:

a21: a22: a23:

و حقلي ادخال للأزمنة



Transfer Times (t):

Line 1:

t11: t12:

Line 2:

t21: t22:

Assembly Line Scheduling

Number of Stations (n):

5

Enter a number greater than 0.

عند ادخال رقم 5 سنجد ظهور خمس حقول ادخال

Station Times (a):

Line 1:

a11: a12: a13: a14: a15:

Line 2:

a21: a22: a23: a24: a25:

وسيظهر 4 حقول لأزمنة الانتقال

Transfer Times (t):

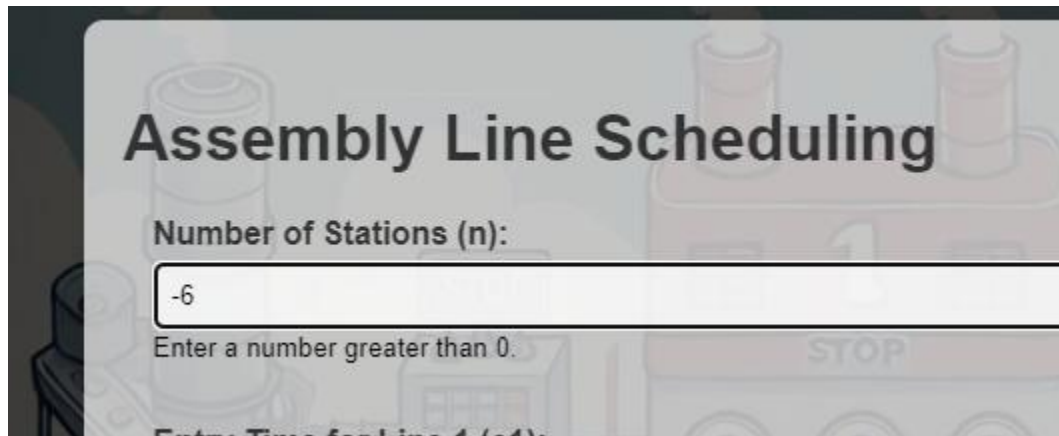
Line 1:

t11: t12: t13: t14:

Line 2:

t21: t22: t23: t24:

في حال ادخل المستخدم عدد محطات اصغر من 0



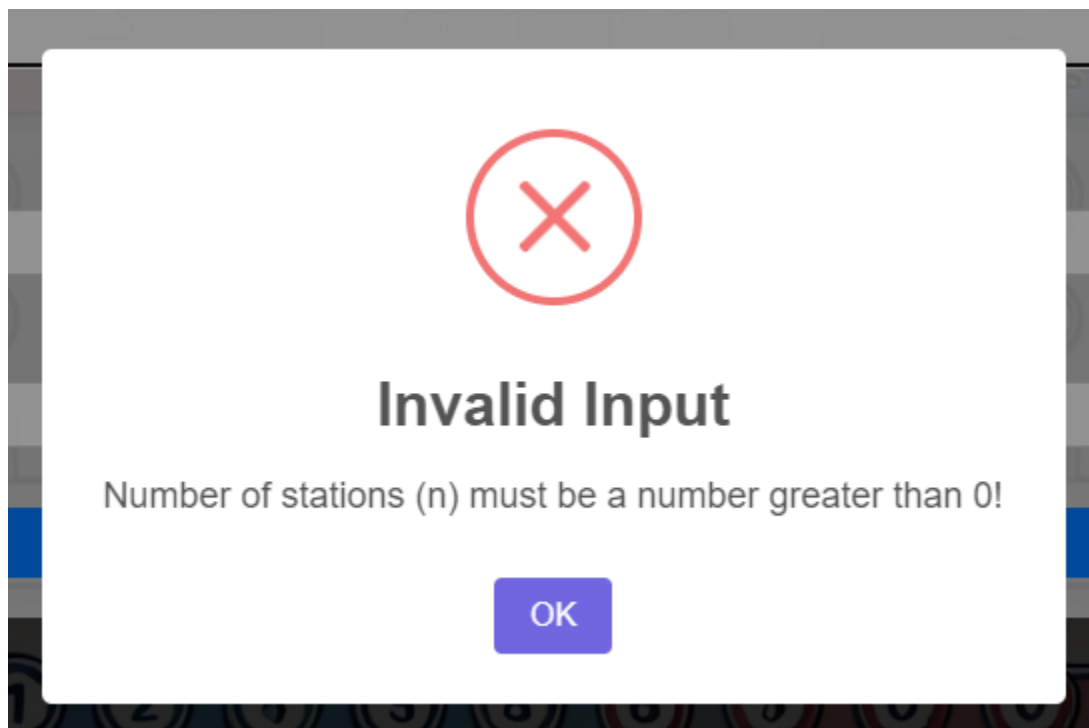
Assembly Line Scheduling

Number of Stations (n):

-6

Enter a number greater than 0.


يظهر التنبيه التالي :



بقية الحقول في حال تم ادخال قيمة سالبة سيظهر تنبيه منبثق صغير بجانب الحقل

Entry Time for Line 1 (e1):

Entry Time for Line 2 (e2):

Exit Time for Line  Value must be greater than or equal to 0.

Exit Time for Line 2 (x2):

Entering Valid Values

Assembly Line Scheduling

Number of Stations (n):
Enter a number greater than 0.

Entry Time for Line 1 (e1):

Entry Time for Line 2 (e2):

Exit Time for Line 1 (x1):

Exit Time for Line 2 (x2):

Station Times (a):

Line 1:

a11:	a12:	a13:	a14:	a15:
<input type="text" value="5"/>	<input type="text" value="8"/>	<input type="text" value="2"/>	<input type="text" value="4"/>	<input type="text" value="3"/>

Line 2:

a21:	a22:	a23:	a24:	a25:
<input type="text" value="6"/>	<input type="text" value="8"/>	<input type="text" value="1"/>	<input type="text" value="2"/>	<input type="text" value="5"/>

Transfer Times (t):

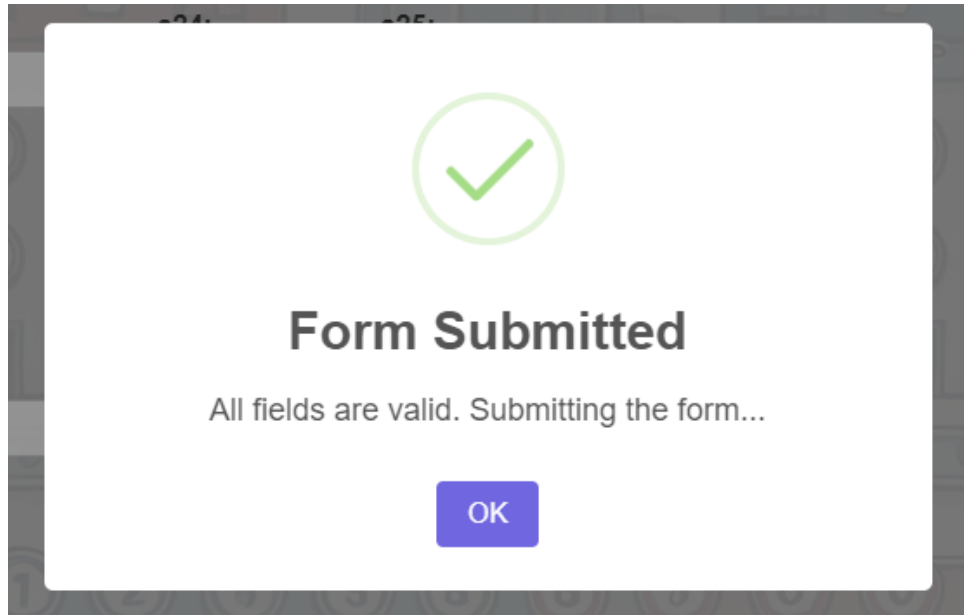
Line 1:

t11:	t12:	t13:	t14:
<input type="text" value="1"/>	<input type="text" value="2"/>	<input type="text" value="2"/>	<input type="text" value="1"/>

Line 2:

t21:	t22:	t23:	t24:
<input type="text" value="3"/>	<input type="text" value="2"/>	<input type="text" value="1"/>	<input type="text" value="1"/>

بعد الضغط على زر حساب سيقوم بحساب الطريق الأمثل عن طريق خوارزمية Assembly
Line Scheduling بالطريقة الديناميكية
سيتم شرحها في قسم الباكت ايند



ظهر اشعار بأن الحقول كلها صحيحة ويتم تسجيل النموذج
وبعدها سيتم الانتقال لصفحة النتائج لعرض النتائج

Result

The screenshot shows a 'Result' window with a light gray background and a dark border. The title 'Result' is centered at the top. Below the title, the text 'Optimal Time = 25' and 'Exit x2 = 1' is displayed. The section 'Optimal Path:' is followed by a list of steps in a white box with a light gray border. The steps are: 'Line 2, Station 5, Time = 5', 'Line 2, Station 4, Time = 2', 'Line 2, Station 3, Time = 1', 'Line 2, Station 2, Time = 8', 'Transfer from Line 2 to Line 1, Time = 3', 'Line 1, Station 1, Time = 5', and 'Entry e1 = 2'. A blue 'Back' button is located at the bottom left of the window.

Result

Optimal Time = 25
Exit x2 = 1

Optimal Path:

- Line 2, Station 5, Time = 5
- Line 2, Station 4, Time = 2
- Line 2, Station 3, Time = 1
- Line 2, Station 2, Time = 8
- Transfer from Line 2 to Line 1, Time = 3
- Line 1, Station 1, Time = 5
- Entry e1 = 2

Back

صفحة النتائج بعد القيام بإدخال قيم المحطات من ناحية ازمدة الانتقال وتكلفة كل محطة
تم حساب الطريق الأمثلي عبر الطريقة الديناميكية

حيث نجد الوقت الأمثلي هو 25 والخروج عبر الخط الثاني والمسار كالاتي :

الخط الثاني المحطة الخامسة ثم الخط الثاني المحطة الرابعة ثم الخط الثاني المحطة الثانية ثم ننتقل
للخط الأول ثم المحطة الأولى في الخط الأول و الدخول من الخط الأول
طبعا يمكن قراءة المسار من الجهتين اما من الخروج للدخول او العكس

Back end

Programming Language : Python

تم اختيار لغة البرمجة بايثون لأنها تُعتبر واحدة من أسهل لغات البرمجة من حيث التعلم والاستخدام. تتميز بتركيبة بسيطة وواضحة تشبه اللغة الإنجليزية، مما يجعلها مثالية للمبتدئين والمطورين على حد سواء. بالإضافة إلى ذلك، بايثون تدعم مجموعة ضخمة من المكتبات التي تغطي مجالات متعددة مثل الذكاء الاصطناعي، تحليل البيانات، وتطوير الويب، مما يجعلها لغة متعددة الاستخدامات. كما أنها تعمل على أنظمة تشغيل مختلفة مثل Windows وLinux وmacOS، مما يوفر مرونة في العمل. أخيرًا، بايثون لديها مجتمع دعم قوي، حيث يمكن للمطورين العثور على مساعدة بسهولة عبر الإنترنت.

Framework : Flask

تم اختيار فلاسك كإطار عمل لتطوير الويب لأنه خفيف الوزن وسهل الاستخدام. يعتبر فلاسك مثاليًا للتطبيقات الصغيرة والمتوسطة حيث لا يتطلب تعقيدات كبيرة في الإعداد. يتميز بمرونته، حيث يمكن إضافة المكونات حسب الحاجة دون الحاجة إلى استخدام كل ميزات الإطار، مما يجعله سهل التخصيص. بالإضافة إلى ذلك، فلاسك يدعم بناء واجهات برمجة التطبيقات (APIs) بسهولة، مما يجعله خيارًا ممتازًا لتطوير تطبيقات ويب تفاعلية. وأخيرًا، لأنه مكتوب بلغة بايثون، فإنه يتكامل بسلاسة مع المكتبات والأدوات الأخرى الخاصة بالبايثون.

Hosting : PythonAnywhere

تم اختيار PythonAnywhere كمنصة استضافة لأنها توفر بيئة مثالية لتطبيقات بايثون وفلاسك. تتميز بسهولة النشر، حيث يمكن نشر التطبيقات بخطوات بسيطة دون الحاجة إلى إعدادات معقدة، مما يوفر الوقت والجهد. كما أنها توفر نسخة تجريبية مجانية تسمح للمطورين

باختبار تطبيقاتهم قبل الالتزام بخطة مدفوعة. تدعم PythonAnywhere قواعد البيانات مثل MySQL، مما يجعلها مناسبة للتطبيقات التي تحتاج إلى تخزين بيانات. بالإضافة إلى ذلك، توفر المنصة دعمًا فنيًا للمستخدمين في حالة وجود أي مشاكل، مما يجعلها خيارًا موثوقًا لتشغيل التطبيقات.

Project Structure

app.py

تتضمن الملف الرئيسي للمشروع حيث يحتوي بداخله الخوارزمية الديناميكية التي قمنا من خلالها بحل مشكلة Assembly Line Scheduling و يحتوي أيضا على المسارات (صفحات الموقع كلها)

```
1 from flask import Flask, render_template, request
2 app = Flask(__name__)
3 def assembly_line_scheduling(n, a, t, e, x):
4     """
5     Calculate the optimal time for assembly line scheduling.
6
7     :param n: Number of stations.
8     :param a: List of station times (a[0] for Line 1, a[1] for Line 2).
9     :param t: List of transfer times (t[0] for Line 1, t[1] for Line 2).
10    :param e: List of entry times (e[0] for Line 1, e[1] for Line 2).
11    :param x: List of exit times (x[0] for Line 1, x[1] for Line 2).
12    :return: Optimal time, exit line, and path details.
13    """
14    # Initialize arrays
15    f1 = [0] * n
16    f2 = [0] * n
17    l1 = [0] * n
18    l2 = [0] * n
19    # Entry times
20    f1[0] = e[0] + a[0][0]
21    f2[0] = e[1] + a[1][0]
22    # Calculate optimal time for each station
23    for j in range(1, n):
24        # Line 1
25        if f1[j-1] + a[0][j] <= f2[j-1] + t[1][j-1] + a[0][j]:
26            f1[j] = f1[j-1] + a[0][j]
27            l1[j] = 1
28        else:
29            f1[j] = f2[j-1] + t[1][j-1] + a[0][j]
30            l1[j] = 2
31        # Line 2
32        if f2[j-1] + a[1][j] <= f1[j-1] + t[0][j-1] + a[1][j]:
33            f2[j] = f2[j-1] + a[1][j]
34            l2[j] = 2
35        else:
36            f2[j] = f1[j-1] + t[0][j-1] + a[1][j]
37            l2[j] = 1
38    # Calculate final optimal time
39    if f1[-1] + x[0] <= f2[-1] + x[1]:
40        f_opt = f1[-1] + x[0]
41        l_opt = 1
42    else:
43        f_opt = f2[-1] + x[1]
44        l_opt = 2
```

```
1 # Build the optimal path with transfer details
2 path = []
3 l = l_opt
4 for j in range(n-1, 0, -1):
5     path.append(f"Line {l}, Station {j+1}, Time = {a[l-1][j]}")
6     if l == 1:
7         if l1[j] == 2:
8             path.append(
9                 f"Transfer from Line 1 to Line 2, Time = {t[0][j-1]}")
10            l = l1[j]
11        else:
12            if l2[j] == 1:
13                path.append(
14                    f"Transfer from Line 2 to Line 1, Time = {t[1][j-1]}")
15            l = l2[j]
16    path.append(f"Line {l}, Station 1, Time = {a[l-1][0]}")
17    path.append(f"Entry e{l} = {e[l-1]}")
18    return f_opt, l_opt, path
19# Route for the home page
20@app.route('/')
21def home():
22    return render_template('home.html')
23# Route for the input page
24@app.route('/input', methods=['GET', 'POST'])
25def input_page():
26    if request.method == 'POST':
27        n = int(request.form['n'])
28        e1 = int(request.form['e1'])
29        e2 = int(request.form['e2'])
30        x1 = int(request.form['x1'])
31        x2 = int(request.form['x2'])
32        a = [[0] * n for _ in range(2)]
33        t = [[0] * (n-1) for _ in range(2)]
34        for i in range(2):
35            for j in range(n):
36                a[i][j] = int(request.form[f'a{i}{j}'])
37            for i in range(2):
38                for j in range(n-1):
39                    t[i][j] = int(request.form[f't{i}{j}'])
40        f_opt, l_opt, path = assembly_line_scheduling(
41            n, a, t, [e1, e2], [x1, x2])
42        return render_template('result.html', f_opt=f_opt, l_opt=l_opt, path=path, x1=x1, x2=x2)
43    return render_template('input.html', n=n)
44if __name__ == '__main__':
45    app.run(debug=True)
```



```
1 from flask import Flask, render_template, request
2 app = Flask(__name__)
```

دالة `assembly_line_scheduling`



```
1 def assembly_line_scheduling(n, a, t, e, x):
2     """
3     Calculate the optimal time for assembly line scheduling.
4
5     :param n: Number of stations.
6     :param a: List of station times (a[0] for Line 1, a[1] for Line 2).
7     :param t: List of transfer times (t[0] for Line 1, t[1] for Line 2).
8     :param e: List of entry times (e[0] for Line 1, e[1] for Line 2).
9     :param x: List of exit times (x[0] for Line 1, x[1] for Line 2).
10    :return: Optimal time, exit line, and path details.
11    """
```

تهيئة المصفوفات



```
1 # Initialize arrays
2 f1 = [0] * n
3 f2 = [0] * n
4 l1 = [0] * n
5 l2 = [0] * n
```

```

1 # Entry times
2     f1[0] = e[0] + a[0][0]
3     f2[0] = e[1] + a[1][0]

```

```

1 # Calculate optimal time for each station
2     for j in range(1, n):
3         # Line 1
4         if f1[j-1] + a[0][j] <= f2[j-1] + t[1][j-1] + a[0][j]:
5             f1[j] = f1[j-1] + a[0][j]
6             l1[j] = 1
7         else:
8             f1[j] = f2[j-1] + t[1][j-1] + a[0][j]
9             l1[j] = 2
10        # Line 2
11        if f2[j-1] + a[1][j] <= f1[j-1] + t[0][j-1] + a[1][j]:
12            f2[j] = f2[j-1] + a[1][j]
13            l2[j] = 2
14        else:
15            f2[j] = f1[j-1] + t[0][j-1] + a[1][j]
16            l2[j] = 1

```

```

1 if f1[-1] + x[0] <= f2[-1] + x[1]:
2     f_opt = f1[-1] + x[0]
3     l_opt = 1
4 else:
5     f_opt = f2[-1] + x[1]
6     l_opt = 2

```

```

1 # Build the optimal path with transfer details
2 path = []
3 l = l_opt
4 for j in range(n-1, 0, -1):
5     path.append(f"Line {l}, Station {j+1}, Time = {a[l-1][j]}")
6     if l == 1:
7         if l1[j] == 2:
8             path.append(
9                 f"Transfer from Line 1 to Line 2, Time = {t[0][j-1]}")
10            l = l1[j]
11        else:
12            if l2[j] == 1:
13                path.append(
14                    f"Transfer from Line 2 to Line 1, Time = {t[1][j-1]}")
15            l = l2[j]
16 path.append(f"Line {l}, Station 1, Time = {a[l-1][0]}")
17 path.append(f"Entry e{1} = {e[l-1]}")
18 return f_opt, l_opt, path

```

مسار الصفحة الرئيسية

```

1 @app.route('/')
2 def home():
3     return render_template('home.html')

```

مسار صفحة الإدخال

```

1 # Route for the input page
2 @app.route('/input', methods=['GET', 'POST'])

```

دالة `input_page`

التحقق من طريقة الطلب (Request Method)

```

1 if request.method == 'POST':

```

يتم التحقق من أن الطلب المرسل من النموذج هو من نوع POST، أي أن المستخدم قام بإرسال البيانات.

```

1 n = int(request.form['n'])
2     e1 = int(request.form['e1'])
3     e2 = int(request.form['e2'])
4     x1 = int(request.form['x1'])
5     x2 = int(request.form['x2'])

```

تم استخراج البيانات المدخلة من النموذج :

n : عدد المحطات.

$e1, e2$: أوقات الدخول للخطين 1 و 2.

$x1, x2$: أوقات الخروج للخطين 1 و 2.

يتم تحويل هذه البيانات إلى أعداد صحيحة باستخدام `int()`.

```

1 a = [[0] * n for _ in range(2)]
2     t = [[0] * (n-1) for _ in range(2)]

```

- يتم إنشاء مصفوفتين:
 - a مصفوفة ثنائية الأبعاد ($2 \times n$) لتخزين أوقات المحطات لكل خط.
 - t مصفوفة ثنائية الأبعاد ($2 \times (n-1)$) لتخزين أوقات النقل بين المحطات.

تعبئة مصفوفات أوقات المحطات

```

1 for i in range(2):
2     for j in range(n):
3         a[i][j] = int(request.form[f'a{i}{j}'])

```

- يتم تعبئة مصفوفة a بقيم أوقات المحطات المدخلة من النموذج.
- يتم استخدام حلقتين متداخلتين:
 - الحلقة الخارجية (i) للتكرار على الخطين (0 و 1).
 - الحلقة الداخلية (j) للتكرار على المحطات (من 0 إلى $n-1$).
- يتم استخراج القيم من النموذج باستخدام `request.form[f'a{i}{j}']`، حيث i و j تمثلان موقع القيمة في المصفوفة.

تعبئة مصفوفات أوقات النقل

```

1 for i in range(2):
2     for j in range(n-1):
3         t[i][j] = int(request.form[f't{i}{j}'])

```

- يتم تعبئة مصفوفة t بقيم أوقات النقل المدخلة من النموذج.
- يتم استخدام حلقتين متداخلتين:
 - الحلقة الخارجية (i) للتكرار على الخطتين (0 و 1).
 - الحلقة الداخلية (j) للتكرار على أوقات النقل بين المحطات (من 0 إلى $n-2$).
- يتم استخراج القيم من النموذج باستخدام `request.form[f't{i}{j}']`.

استدعاء دالة `assembly_line_scheduling`

```

1 f_opt, l_opt, path = assembly_line_scheduling(n, a, t, [e1, e2], [x1, x2])

```

- يتم استدعاء الدالة `assembly_line_scheduling` لحساب الوقت الأمثل (f_opt)، الخط الأمثل (l_opt)، ومسار العملية ($path$).
- يتم تمرير البيانات المدخلة ($n, a, t, e1, e2, x1, x2$) كمعاملات للدالة.

عرض النتائج :

```

1 return render_template('result.html', f_opt=f_opt, l_opt=l_opt, path=path, x1=x1, x2=x2)

```

- يتم عرض النتائج على صفحة `result.html` باستخدام دالة `render_template`.
- يتم تمرير المتغيرات التالية إلى الصفحة:
 - f_opt : الوقت الأمثل.
 - l_opt : الخط الأمثل.
 - $path$: تفاصيل المسار.
 - $x2, x1$: أوقات الخروج.

عرض صفحة الإدخال (إذا لم يتم إرسال النموذج)

```

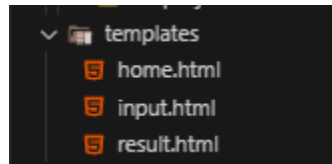
1 return render_template('input.html', n=0)

```

- إذا لم يتم إرسال النموذج (أي أن الطلب ليس من نوع POST)، يتم عرض صفحة الإدخال (`input.html`) مع تمرير القيمة $n=0$ لتهيئة الصفحة.

```
1 if __name__ == '__main__':
2     app.run(debug=True)
```

templates/



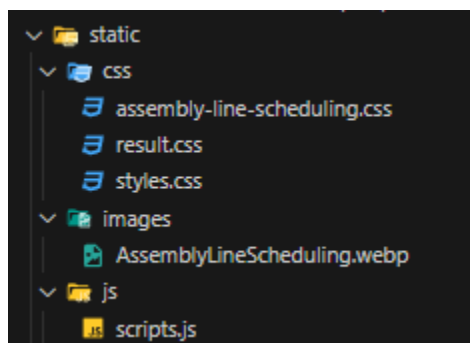
يحتوي ضمنه على ثلاث ملفات Html تم التحديث عنهم في قسم الفرونت ايند
home.html input.html result.html

home.html هي الصفحة الرئيسية للموقع

input.html هي الصفحة التي يتم ادخال البيانات ضمنها

result.html هي الصفحة التي يتم عرض النتائج بعد تطبيق الخوارزمية

static/



يحتوي على باقي ملفات المشروع

Css : التي تحتوي على ملفات ال css التي تنسق شكل الصفحات

Images : تحتوي على خلفية الموقع

Js : يحتوي على ملف scripts.js الذي يقوم ببعض التحقق من الحقول

شرح ملف *scripts*

دالة **validateNumber**

```
1 function validateNumber () {
2   const nInput = document.getElementById('n')
3   const nValue = parseInt(nInput.value)
4
5   if (nValue <= 0 || isNaN(nValue)) {
6     Swal.fire({
7       icon: 'error',
8       title: 'Invalid Input',
9       text: 'Number of stations (n) must be a number greater than 0!'
10    })
11    nInput.value = '' // Clear the input field
12    nInput.focus() // Focus back on the input field
13    return false
14  }
15  return true
16}
```

- تقوم هذه الدالة بالتحقق من صحة قيمة عدد المحطات (n).
- إذا كانت القيمة غير صالحة (أقل من أو تساوي صفر أو ليست رقمًا)، يتم عرض رسالة خطأ باستخدام مكتبة **SweetAlert2** وإعادة التركيز على حقل الإدخال.

```

1 function updateFields () {
2   const n = parseInt(document.getElementById('n').value)
3   const aFields = document.getElementById('a-fields')
4   const tFields = document.getElementById('t-fields')
5
6   // Clear previous fields
7   aFields.innerHTML = ''
8   tFields.innerHTML = ''
9
10  if (n > 0) {
11    // Add station times (a) fields
12    aFields.innerHTML += '<h2>Station Times (a):</h2>'
13    for (let i = 0; i < 2; i++) {
14      aFields.innerHTML += '<h3>Line ${i + 1}:</h3>'
15      let row = '<div class="input-row">'
16      for (let j = 0; j < n; j++) {
17        row += `
18          <div class="form-group">
19            <label for="a${i}${j}">a${i + 1}${j + 1}:</label>
20            <input type="number" id="a${i}${j}" name="a${i}${j}" required min="0">
21          </div>`
22      }
23      row += '</div>'
24      aFields.innerHTML += row
25    }
26
27    // Add transfer times (t) fields
28    tFields.innerHTML += '<h2>Transfer Times (t):</h2>'
29    for (let i = 0; i < 2; i++) {
30      tFields.innerHTML += '<h3>Line ${i + 1}:</h3>'
31      let row = '<div class="input-row">'
32      for (let j = 0; j < n - 1; j++) {
33        row += `
34          <div class="form-group">
35            <label for="t${i}${j}">t${i + 1}${j + 1}:</label>
36            <input type="number" id="t${i}${j}" name="t${i}${j}" required min="0">
37          </div>`
38      }
39      row += '</div>'
40      tFields.innerHTML += row
41    }
42  }
43}

```

- تقوم هذه الدالة بتحديث حقول الإدخال بشكل ديناميكي بناءً على عدد المحطات (n).
- يتم إنشاء حقول إدخال لأوقات المحطات (a) وأوقات النقل (t) لكل خط.
- يتم مسح الحقول القديمة قبل إنشاء الحقول الجديدة.

```

1 function validateForm (e) {
2   e.preventDefault() // Prevent form submission
3
4   const n = parseInt(document.getElementById('n').value)
5   const e1 = document.getElementById('e1').value
6   const e2 = document.getElementById('e2').value
7   const x1 = document.getElementById('x1').value
8   const x2 = document.getElementById('x2').value
9
10  // Validate number of stations (n)
11  if (n <= 0 || isNaN(n)) {
12    Swal.fire({
13      icon: 'error',
14      title: 'Invalid Input',
15      text: 'Number of stations (n) must be a number greater than 0!'
16    })
17    return
18  }
19
20  // Validate entry and exit times
21  if (
22    !e1 ||
23    !e2 ||
24    !x1 ||
25    !x2 ||
26    isNaN(e1) ||
27    isNaN(e2) ||
28    isNaN(x1) ||
29    isNaN(x2)
30  ) {
31    Swal.fire({
32      icon: 'error',
33      title: 'Invalid Input',
34      text: 'Entry and Exit times must be filled with valid numbers!'
35    })
36    return
37  }
38
39  // Validate station times (a) and transfer times (t)
40  let isValid = true
41  let errorMessage = ''
42
43  // Check station times (a)
44  for (let i = 0; i < 2; i++) {
45    for (let j = 0; j < n; j++) {
46      const aInput = document.getElementById(`a${i}${j}`)
47      if (!aInput || !aInput.value || isNaN(aInput.value) || aInput.value < 0) {
48        isValid = false
49        errorMessage = `Station time a${i + 1}${j + 1} is missing or invalid!`
50        break
51      }
52    }
53    if (!isValid) break
54  }
55
56  // Check transfer times (t)
57  if (isValid) {
58    for (let i = 0; i < 2; i++) {
59      for (let j = 0; j < n - 1; j++) {
60        const tInput = document.getElementById(`t${i}${j}`)
61        if (
62          !tInput ||
63          !tInput.value ||
64          isNaN(tInput.value) ||
65          tInput.value < 0
66        ) {
67          isValid = false
68          errorMessage = `Transfer time t${i + 1}${j + 1} is missing or invalid!`
69          break
70        }
71      }
72    }
73    if (!isValid) break
74  }
75
76  // Show error message if any field is invalid
77  if (!isValid) {
78    Swal.fire({
79      icon: 'error',
80      title: 'Invalid Input',
81      text: errorMessage
82    })
83    return
84  }
85
86  // If all fields are valid, submit the form
87  Swal.fire({
88    icon: 'success',
89    title: 'Form Submitted',
90    text: 'All fields are valid. Submitting the form...'
91  }).then(() => {
92    document.getElementById('input-form').submit()
93  })
94
95 }
96

```

- تقوم هذه الدالة بالتحقق من صحة جميع الحقول قبل إرسال النموذج.
- يتم التحقق من:
 - عدد المحطات (n).
 - أوقات الدخول (e1, e2) والخروج (x1, x2).
 - أوقات المحطات (a) وأوقات النقل (t).
- إذا كانت جميع الحقول صالحة، يتم إرسال النموذج. وإلا، يتم عرض رسالة خطأ.

```
1 document.getElementById('input-form').addEventListener('submit', validateForm)
```

- يتم إرفاق دالة validateForm بحدث submit للنموذج، بحيث يتم تنفيذ التحقق قبل الإرسال.

Assembly Line Scheduling وصف الخوارزمية

Assembly Line Scheduling هو مشكلة كلاسيكية في مجال الخوارزميات والبرمجة الديناميكية

(Dynamic Programming). تهدف هذه المشكلة إلى تحسين عملية التصنيع في مصنع يحتوي على خطوط إنتاج متعددة، حيث يتم نقل

المنتجات عبر سلسلة من المحطات في كل خط إنتاج. الهدف هو تحديد المسار الأمثل لنقل المنتج عبر هذه المحطات بحيث يتم تقليل الوقت الإجمالي أو التكلفة الإجمالية للإنتاج.

وصف المشكلة:

1. **خطوط الإنتاج:** يوجد خطان إنتاج (أو أكثر) متوازيان، كل خط يحتوي على عدد من المحطات (Stations).
2. **المحطات:** كل محطة في الخط تقوم بعملية معينة على المنتج. الوقت اللازم لإكمال العملية في كل محطة معروف.
3. **نقاط النقل:** يمكن نقل المنتج من خط إنتاج إلى آخر في نقاط محددة بين المحطات، ولكن هذا النقل يتطلب وقتًا إضافيًا (Transfer Time).
4. **البداية والنهاية:** يبدأ المنتج في بداية أحد الخطوط وينتهي في نهاية أحد الخطوط.

الهدف:

تحديد المسار الذي يقلل الوقت الإجمالي اللازم لإكمال تصنيع المنتج، مع الأخذ في الاعتبار الوقت في كل محطة ووقت النقل بين الخطوط.

حل المشكلة باستخدام البرمجة الديناميكية:

يمكن حل هذه المشكلة باستخدام البرمجة الديناميكية عن طريق تقسيم المشكلة إلى مشكلات فرعية أصغر وحل كل مشكلة فرعية مرة واحدة فقط، ثم تخزين النتائج لاستخدامها لاحقًا.

الكود :

1.دالة assembly_line_scheduling:

هذه الدالة هي الجزء الأساسي من الخوارزمية، حيث يتم فيها حساب الوقت الأمثل لتصنيع المنتج عبر خطوط الإنتاج. يتم تلخيص وظيفتها كالتالي:

المدخلات:

- n: عدد المحطات في كل خط إنتاج.
- a: قائمة تحتوي على الأوقات اللازمة في كل محطة لكل خط إنتاج. [0]a للأوقات في الخط 1، و [1]a للأوقات في الخط 2.

- t: قائمة تحتوي على الأوقات اللازمة للنقل بين الخطوط. [o]t للأوقات للنقل من الخط 1 إلى الخط 2، و [1]t للنقل من الخط 2 إلى الخط 1.

- e: قائمة تحتوي على الأوقات اللازمة لدخول كل خط إنتاج. [o]e للدخول إلى الخط 1، و [1]e للدخول إلى الخط 2.
- x: قائمة تحتوي على الأوقات اللازمة للخروج من كل خط إنتاج. [o]x للخروج من الخط 1، و [1]x للخروج من الخط 2.

المخرجات:

- f_opt: الوقت الأمثل الإجمالي لإكمال التصنيع.
- l_opt: الخط الذي يتم الخروج منه (1 أو 2).
- path: المسار الأمثل مع تفاصيل النقل بين الخطوط.

خطوات الدالة:

1. تهيئة المصفوفات:

- يتم إنشاء مصفوفتين f1 و f2 لتخزين الوقت الأمثل للوصول إلى كل محطة في الخط 1 والخط 2 على التوالي.
- يتم إنشاء مصفوفتين l1 و l2 لتخزين الخط الذي تم الانتقال منه للوصول إلى كل محطة.

2. حساب الوقت لدخول أول محطة:

- يتم حساب الوقت لدخول الخط 1 وإكمال المحطة الأولى باستخدام العلاقة: $[o][f1[o]] = e[o] + a[o]$.
- يتم حساب الوقت لدخول الخط 2 وإكمال المحطة الأولى باستخدام العلاقة: $[o][f2[o]] = e[1] + a[1]$.

3. حساب الوقت الأمثل لكل محطة:

- يتم حساب الوقت الأمثل لكل محطة باستخدام العلاقات التكرارية:
- $$f1[j] = \min(f1[j-1] + a[o][j], f2[j-1] + t[1][j-1] + a[o][j])$$
- $$f2[j] = \min(f2[j-1] + a[1][j], f1[j-1] + t[o][j-1] + a[1][j])$$
- يتم تحديث l1 و l2 لتتبع الخط الذي تم الانتقال منه.

4. حساب الوقت الأمثل الإجمالي:

- يتم حساب الوقت الأمثل الإجمالي باستخدام العلاقة: $f_opt = \min(f1[-1] + x[o], f2[-1] + x[1])$.

5. بناء المسار الأمثل:

- يتم تتبع المسار الأمثل بدءًا من المحطة الأخيرة وحتى المحطة الأولى.
- يتم تسجيل تفاصيل النقل بين الخطوط إذا حدثت.

نهاية التقرير