Datrysiadau i Daflen Problemau 4

1. Defnyddiwch efelychiad Monte Carlo i amcangyfrif gwerth yr integryn canlynol, gan rhoi cyfwng hyder ar gyfer eich amcangyfrif.

$$I = \int_{3/2}^{5/2} x^3 + 1 \, dx$$

Datrysiad 1 *Yn gyntaf crëwn ffwythiant ar gyfer amcangyfrif cyfrannau:*

```
>>> def amcangyfrif_cyfran(integrand, xisaf, xuchaf, ymax, N):
...     n = 0
...     for trial in range(N):
...         x = ((xuchaf - xisaf) * random.random()) + xisaf
...         y = random.random() * ymax
...         if y < integrand(x):
...             n += 1
...     return n / N
```

Nawr, ffwythiant ar gyfer cyfrifo cyfyngau hyder

```
>>> def cyfwng_hyder(p, N, arwynebedd):
...     isaf = p - 1.96 * ((p * (1 - p) / N) ** 0.5)
...     uchaf = p + 1.96 * ((p * (1 - p) / N) ** 0.5)
...     return (arwynebedd * isaf, arwynebedd * uchaf)
```

Nawr diffiniwn yr integrand:

```
>>> def integrand_1(x):
...     return (x ** 3) + 1
```

Mae hwn yn ffwythiant sy'n cynyddu, felly'r gwerth mwyaf gall yr integrand cymryd yw pan mae $x = 5/2$. Felly samplwn o fewn petryal gydag uchder o 0 i'r uchafbwynt hon, a lled o $3/2$ i $5/2$.

Datrysiad 1 (continuing from p. 1) *Yna:*

```
>>> random.seed(0)
>>> N = 1000
>>> xisaf, xuchaf, ymax = 3/2, 5/2, integrand_1(5/2)
>>> arwynebedd = (xuchaf - xisaf) * ymax
>>> p = amcangyfrif_cyfran(integrand_1, xisaf, xuchaf, ymax, N)
>>> I = p * arwynebedd
>>> I
9.559375
>>> cyfwng_hyder(p, N, arwynebedd)
(9.049990046124861, 10.068759953875137)
```

Felly gallwn cael 95% hyder fod $9.05 < I < 10.07$.

2. Defnyddiwch efelychiad Monte Carlo i amcangyfrif gwerth yr integryn canlynol, gan rhoi cyfwng hyder ar gyfer eich amcangyfrif.

$$I = \int_{-1}^{1} \sqrt{1 - x^2}\, dx$$

Datrysiad 2 *Diffiniwn yr integrand:*

```
>>> def integrand_2(x):
...     return (1 - (x ** 2)) ** 0.5
```

Dyma hanner cylch uchaf radiws 1, ac felly gwerth mwyaf yr integrand yw 1. Gan ddefnyddio'r un ffwythiannau a'r cwestiwn blaenorol, samplwn pwyntiau yn y petryal uchder [0, 1], a lled [-1, 1]:

```
>>> random.seed(0)
>>> N = 1000
>>> xisaf, xuchaf, ymax = -1, 1, 1
>>> arwynebedd = (xuchaf - xisaf) * ymax
>>> p = amcangyfrif_cyfran(integrand_2, xisaf, xuchaf, ymax, N)
>>> I = p * arwynebedd
>>> I
1.542
>>> cyfwng_hyder(p, N, arwynebedd)
(1.4899128017877714, 1.5940871982122287)
```

Felly gallwn cael 95% hyder fod $1.49 < I < 1.59$.

3. Defnyddiwch efelychiad Monte Carlo er mwyn canfod y tebygolrwydd p, pan taflir pedwar dis ar yr un pryd, y gallant cael eu holltu i mewn i dwy bâr gyda symiau hafal. Er enghraifft, mae'r tafliad $(3, 1, 3, 5)$ yn gallu cael ei holltu i mewn i dwy bâr gyda symiau hafal: $3+3 = 1+5$, ond ni all $(2, 2, 2, 4)$. Rhowch cyfwng hyder ar gyfer eich amcangyfrif.

Datrysiad 3 *Bydd dau ffwythiant yn defnyddiol, yn gyntaf, ffwythiant ar gyfer taflu dis:*

```
>>> def taflu():
...     return random.randint(1, 6)
```

yna, ffwythiant i wirio os gallwn holltu pedwar harif i mewn i ddwy bâr gyda symiau hafal:

```
>>> def gwirio(a, b, c, d):
...     cyf1 = (a + b) == (c + d)
...     cyf2 = (a + c) == (b + d)
...     cyf3 = (a + d) == (b + c)
...     if cyf1 or cyf2 or cyf3:
...         return True
```

Nawr rhedwn yr efelychiad gyda $N = 1000$ o dreialon:

```
>>> random.seed()
>>> N = 1000
>>> n = 0
>>> for treial in range(N):
...     a, b, c, d = taflu(), taflu(), taflu(), taflu()
...     if gwirio(a, b, c, d):
...         n += 1
>>> p = n / N
>>> p
0.286
```

ac yn olaf y cyfwng hyder:

```
>>> isaf = p - 1.96 * ((p * (1 - p) / N) ** 0.5)
>>> uchaf = p + 1.96 * ((p * (1 - p) / N) ** 0.5)
>>> (isaf, uchaf)
(0.2579916068579434, 0.31400839314205653)
```

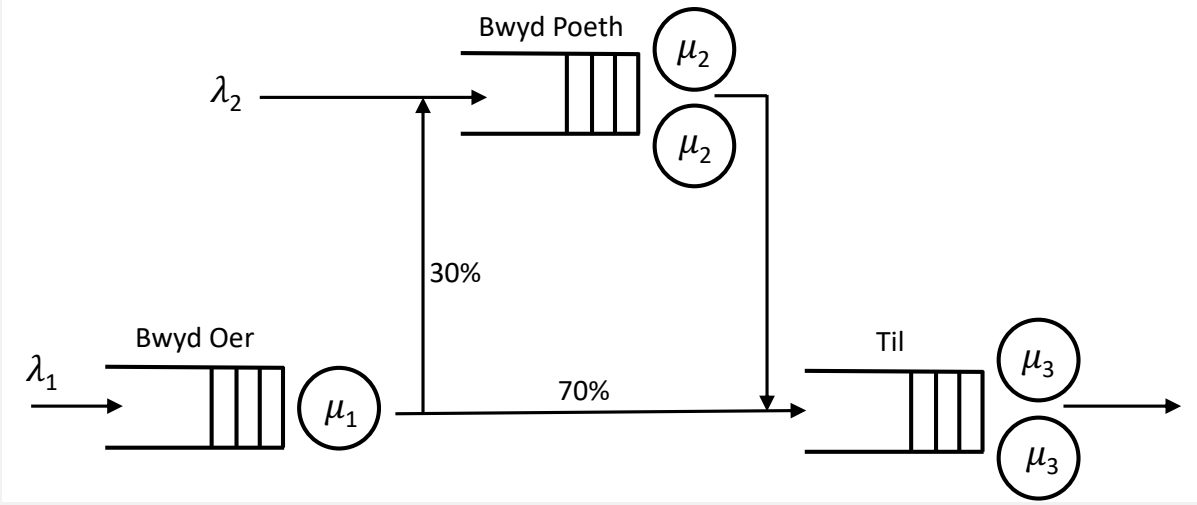
Felly gallwn cael 95% hyder fod $0.258 < p < 0.314$.

4. Mae'r system ar gyfer archebu bwyd mewn caffi yng nghanol Caerdydd yn ymddwyn fel rhwydwaith o giwiau. Mae tri cownter: y cownter bwyd oed, y cownter bwyd poeth, a'r til lle mae'r cwsmeriaid yn talu am eu bwyd. Os yw'r cwsmer eisiau bwyd oer yn unig, mae'n nhw'n ymuno a'r ciw bwyd oer; os ydynt eisiau bwyd poeth yn unig, mae'n nhw'n ymuno a'r ciw bwyd poeth; os ydynt eisiau bwyd oer a bwyd poeth, mae angen iddyn nhw ymuno a'r cownter bwyd oer cyntaf, ac yna mynd i'r cownter bwyd poeth. Ar ôl pigo lan eu bwyd, mae angen iddyn nhw hefyd ciwio wrth y til i dalu.

   - Mae cwsmeriaid yn cyrraedd i'r cownter bwyd oer ar gyfradd 19 yr awr,
   - Mae cwsmeriaid yn cyrraedd i'r cownter bwyd poeth ar gyfradd 12 yr awr,
   - Mae 30% o gwsmeriaid sy'n ciwio at y cownter bwyd oed hefyd eisiau bwyd poeth,
   - Ar gyfartaledd mae'n cymryd 1 munud i weini bwyd oer, 2 a hanner munud i weini bwyd poeth, a 2 munud i dalu,
   - Mae 1 gweinydd wrth y cownter bwyd oer, 2 weinydd wrth y cownter bwyd poerth, a 2 weinydd wrth y til,
   - Mae'r caffi ar agor am 3 awr yn ystod yr amser cinio.

   Mae'r caffi eisiau gwybod faint o gwsmeriaid byddant yn disgwyl gweini pob amser cinio ar gyfartaledd. Perfformiwch efelychiad digwyddiad-arwahanol gyda Ciw, a dangoswch y model cysyniadol.

Datrysiad 4 *Rhoddir y model cysyniadol gan y diagram isod:*



Tybiwn fod dyfodiadau yn Markovaidd a bod amseroedd gwasanaeth wedi'i dosrannu'n Esbonyddol. Gan ddewis ein unedau amser fel munudad, cawn $\lambda_1 = {}^{18}/_{60} = 0.3$, $\lambda_2 = {}^{12}/_{60} = 0.2$, $\mu_1 = {}^{1}/_{1} = 1$, $\mu_2 = {}^{1}/_{2.5} = 0.4$, *a* $\mu_3 = {}^{1}/_{2} = 0.5$.

Datrysiad 4 (continuing from p. 4) *Felly y rhwydwaith Ciw bydd:*

```
>>> import ciw
>>> N = ciw.create_network(
...     arrival_distributions=[ciw.dists.Exponential(rate=0.3),
...                            ciw.dists.Exponential(rate=0.2),
...                            None],
...     service_distributions=[ciw.dists.Exponential(rate=1.0),
...                            ciw.dists.Exponential(rate=0.4),
...                            ciw.dists.Exponential(rate=0.5)],
...     routing=[[0.0, 0.3, 0.7],
...              [0.0, 0.0, 1.0],
...              [0.0, 0.0, 0.0]],
...     number_of_servers=[1, 2, 2]
... )
```

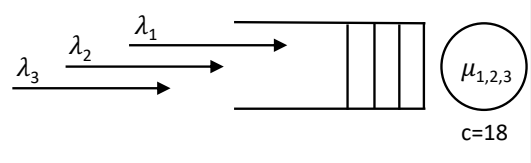
Mae hwn yn broblem terfynus, gan fod y system yn dechrau o gwag pob amser cinio. Felly mae angen rhedeg treialon. Yn dewis 10 treial:

```
>>> cwsmeriaid_wedi_gorffen = []
>>> for treial in range(10):
...     ciw.seed(treial)
...     Q = ciw.Simulation(N)
...     Q.simulate_until_max_time(180)
...     recs = Q.get_all_records()
...     nifer_wedi_gorffen = len([r for r in recs if r.node==3])
...     cwsmeriaid_wedi_gorffen.append(nifer_wedi_gorffen)

>>> N = sum(cwsmeriaid_wedi_gorffen) / len(cwsmeriaid_wedi_gorffen)
80.4
```

5. Mewn adran achosion brys sydd ar agor am 24 awr, mae cleifion yn cyrraedd yn ôl proses Poisson ar gyfradd 35 yr awr. O rhain, caiff $1/7$ eu categoreiddio fel categori Brysbennu 1; $2/7$ fel categori Brysbennu 2; a $4/7$ fel categori Brysbennu 3. Mae cleifion yn aros i gael eu gweld gan un o'r 19 doctor sy'n gwetihio yn yr adran achosion brys, a chaiff eu gweld gan y doctor yn ôl ei flaenoriaethau: Mae gan cleifion Byrsbennu 1 blaenoriaeth dros cleifion Brysbennu 2, sydd yn ei tro yn cael blaenoriaeth dros cleifion Brysbennu 3. Mae'r amser mae'n cymryd i weld pob claf yn dibynnu ar ei categori brysbennu: caif cleifion Brysbennu 3 apwyntiad 15 munud; mae'r amser mae cleifion Brysbennu 2 yn gwario gyda doctor wedi'i dosrannu'n Unffurf rhwng 15 a 25 munud; ac mae'r amser mae cleifion Brysbennu 1 yn gwario gyda doctor wedi'i dosrannu'n Unffurf rhwng 20 a 90 munud. Beth yw'r amser aros cymedrig ar gyfer cleifion ym mhob categori brysbennu?

Datrysiad 5 *Rhoddir y model cysyniadol gan y diagram isod, lle mae saethau dyfodi darwahanedig yn cynrychioli'r blaenoriaethau:*



Gallwn deilio gyda'r gwahanol categorïau brysbennu trwy ystyried gwahanol dosbarthau cwsmer. Gan ddefnyddio munudau fel yr uned amser, mae gennym fod $\lambda_1 = {}^5/_{60}$ *oherwydd teneuo Poisson, ac yn debyd* $\lambda_2 = {}^{10}/_{60}$ *a* $\lambda_3 = {}^{20}/_{60}$.
Felly y rhwydwaith Ciw yw:

```
>>> import ciw
>>> N = ciw.create_network(
...     arrival_distributions={
...         'B1': [ciw.dists.Exponential(rate=5/60)],
...         'B2': [ciw.dists.Exponential(rate=10/60)],
...         'B3': [ciw.dists.Exponential(rate=20/60)]
...     },
...     service_distributions={
...         'B1': [ciw.dists.Uniform(20, 90)],
...         'B2': [ciw.dists.Uniform(15, 25)],
...         'B3': [ciw.dists.Deterministic(value=15)]
...     },
...     number_of_servers=[18],
...     priority_classes={'B1': 0, 'B2': 1, 'B3': 2}
... )
```

yn ddefnyddio'r allweddair `priority_classes` i ddynodi'r blaenoriaethau.
Dyma problem cyflwr-sefydlog, ac felly dewisiwn amser rhediad hir iawn, un mis, ac amser cymhesu digonol, yn diwrnod:

```
>>> ciw.seed(0)
>>> Q = ciw.Simulation(N)
>>> Q.simulate_until_max_time(31 * 24 * 60)
>>> pob_rec = Q.get_all_records()
>>> recs = [r for r in pob_rec if r.arrival_date > (24 * 60)]
```

Datrysiad 5 (continuing from p. 6) *a gallwn canfod yr amser aros cymedrig ar gyfer pob categori brysbennu:*

```
>>> aros_t1 = [r.waiting_time for r in recs if r.customer_class=='B1']
>>> aros_t2 = [r.waiting_time for r in recs if r.customer_class=='B2']
>>> aros_t3 = [r.waiting_time for r in recs if r.customer_class=='B3']
>>> sum(aros_t1) / len(aros_t1)
0.17204197587155948
>>> sum(aros_t2) / len(aros_t2)
0.2501438257279699
>>> sum(aros_t3) / len(aros_t3)
0.6713875498450778
```