[7] Trabajo: Prestaciones de los sistemas

Importación de las librerías necesarias

In [1]:
```
import numpy as np
import utils_ecuaciones as utils
import matplotlib.pyplot as plt
```

Matrices de probabilidad

En este trabajo se condideran dos tipos de usuarios, los usuarios de tipo A y los de tipo B, los cuales utilizan un sistema de subastas web de diferente forma. La matriz de probabilidad para cada tipo de usuario ha sido extraida del capitulo 8 del libro de Menascé.

Table 8.1. Matrix of Transition Probabilities for the CBMG of Type A Sessions

	(e)	(h)	(s)	(v)	(g)	(c)	(b)	(x)
Entry (e)	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
Home (h)	0.00	0.00	0.70	0.00	0.10	0.00	0.00	0.20
Search (s)	0.00	0.00	0.40	0.20	0.15	0.00	0.00	0.25
View Bids (v)	0.00	0.00	0.00	0.00	0.65	0.00	0.00	0.35
Login (g)	0.00	0.00	0.00	0.00	0.00	0.30	0.60	0.10
Create Auction (c)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
Place Bid (b)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
Exit (x)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 8.2. Matrix of Transition Probabilities for the CBMG of Type B Sessions

	(e)	(h)	(s)	(v)	(g)	(c)	(b)	(x)
Entry (e)	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
Home (h)	0.00	0.00	0.70	0.00	0.10	0.00	0.00	0.20
Search (s)	0.00	0.00	0.45	0.15	0.10	0.00	0.00	0.30
View Bids (v)	0.00	0.00	0.00	0.00	0.40	0.00	0.00	0.60
Login (g)	0.00	0.00	0.00	0.00	0.00	0.30	0.55	0.15
Create Auction (c)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
Place Bid (b)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
Exit (x)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Como se puede ver a continuación, hemos creado una variable llamada `matriz_probabilidad_tipo_A`, la cual contiene la matriz de porbabilidad correspondiente al usuario de tipo A y, también, hemos creado otra variable llamada `matriz_probabilidad_tipo_B`, la cual contiene la matriz de porbabilidad correspondiente al usuario de tipo B.

In [2]:
```
matriz_probabilidad_tipo_A = [
    [0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
    [0.0, 0.0, 0.7, 0.0, 0.1, 0.0, 0.0, 0.2],
    [0.0, 0.0, 0.4, 0.2, 0.15, 0.0, 0.0, 0.25],
    [0.0, 0.0, 0.0, 0.0, 0.65, 0.0, 0.0, 0.35],
    [0.0, 0.0, 0.0, 0.0, 0.0, 0.3, 0.6, 0.1],
    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0],
    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0],
    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
]
matriz_probabilidad_tipo_B = [
    [0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
    [0.0, 0.0, 0.7, 0.0, 0.1, 0.0, 0.0, 0.2],
    [0.0, 0.0, 0.45, 0.15, 0.1, 0.0, 0.0, 0.3],
    [0.0, 0.0, 0.0, 0.0, 0.4, 0.0, 0.0, 0.6],
    [0.0, 0.0, 0.0, 0.0, 0.0, 0.3, 0.55, 0.15],
    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0],
    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0],
    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
]


def print_matrix(matrix):
    # printear matriz de manera bonita y de forma igual
    indexes = ["e", "h", "s", "v", "g", "c", "b", "x"]
    print("     (e)   (h)   (s)   (v)   (g)   (c)   (b)   (x)")
    for i, row in enumerate(matrix):
        print(f"({indexes[i]})", end=" ")
        for elem in row:
            print(f"{elem:5.2f}", end=" ")
        print()
    print()


print("Matriz A")
print_matrix(matriz_probabilidad_tipo_A)
print("Matriz B")
print_matrix(matriz_probabilidad_tipo_B)
```

```
Matriz A
     (e)   (h)   (s)   (v)   (g)   (c)   (b)   (x)
(e)  0.00  1.00  0.00  0.00  0.00  0.00  0.00  0.00
(h)  0.00  0.00  0.70  0.00  0.10  0.00  0.00  0.20
(s)  0.00  0.00  0.40  0.20  0.15  0.00  0.00  0.25
(v)  0.00  0.00  0.00  0.00  0.65  0.00  0.00  0.35
(g)  0.00  0.00  0.00  0.00  0.00  0.30  0.60  0.10
(c)  0.00  0.00  0.00  0.00  0.00  0.00  0.00  1.00
(b)  0.00  0.00  0.00  0.00  0.00  0.00  0.00  1.00
(x)  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00

Matriz B
     (e)   (h)   (s)   (v)   (g)   (c)   (b)   (x)
(e)  0.00  1.00  0.00  0.00  0.00  0.00  0.00  0.00
(h)  0.00  0.00  0.70  0.00  0.10  0.00  0.00  0.20
(s)  0.00  0.00  0.45  0.15  0.10  0.00  0.00  0.30
(v)  0.00  0.00  0.00  0.00  0.40  0.00  0.00  0.60
(g)  0.00  0.00  0.00  0.00  0.00  0.30  0.55  0.15
(c)  0.00  0.00  0.00  0.00  0.00  0.00  0.00  1.00
(b)  0.00  0.00  0.00  0.00  0.00  0.00  0.00  1.00
(x)  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
```

Porcentaje de visitas que un de usuario realiza a cada estado

Partiendo de las matrices anteriores, se puede calcular, para cada tipo de usuario, el porcentaje de visitas que realiza a cada estado. Para ello se emplea la siguiente ecuación:

$$v^{\alpha}[i] = \sum_{k \in I} v^{\alpha}[k] * p^{\alpha}_{k,i}$$

Donde

- $i \in I$
- $i \neq (e)$, $I = e, h, s, g, v, c, b, x$
- $\alpha \in A, B$
- $p^{\alpha}_{k,i}$ es la probabilidad de que el tipo usuario α, estando en el estado k, a continaucón realice una nueva operación para visitar el estado i.

Al igual que para las matrices de probabilidad, los valores del porcentaje de visitas que cada tipo de usuario realiza a cada estado se pueden obtener del libro de Menascé.

Table 8.3. Average Visits for Sessions of Type A and B

State	Session A	Session B
V_e	1.000	1.000
V_h	1.000	1.000
V_s	1.167	1.273
V_v	0.233	0.191
V_g	0.427	0.304
V_c	0.128	0.091
V_b	0.256	0.167

Como se puede ver a continuación, hemos creado una variable llamada `v_a`, la cual es un array que contiene el porcentaje de visitas que el usuario de tipo A realiza a cada estado y, también, hemos creado una variable llamada `v_b`, la cual es un array que contiene el porcentaje de visitas que el usuario de tipo B realiza a cada estado.

In [3]:
```
v_a = np.array(
    [
        1.0,
        1.167,
        0.233,
        0.427,
        0.128,
        0.256,
    ]
)


v_b = np.array(
    [
        1.0,
        1.273,
        0.191,
        0.304,
        0.091,
        0.167,
    ]
)
```

Frecuencia de aparición de cada tipo de usuario

Como se puede observar a continuación, hemos creado una variable llamada `f_a` que cotienen la frecuencia de aparición del usuario de tipo A y, también, hemos creado otra variable llamada `f_b` que contiene la frecuencia de apareción del usuario de tipo B.

In [4]:
```
f_a = 0.25
```

```
f_b = 0.75
```

Arquitectura del sistema

La arquitectura física donde se despliega el software del sistema de subastas web se compone de 3 servidores:

- Servidor web
- Sevidor de aplicaciones
- Servidor de base de datos Además, cabe decir que cada servidor cuenta con CPU y DISCO y las diferentes operaciones demandan dichos recursos.

De esta forma, hemos creado una variable `K`, la cual es un array que contiene los recursos existentes en el sistema y, además, hemos creado una variable `R`, la cual contiene el conjunto de las distintas clases de operaciones que consumen recursos de computación.

In [5]:
```
K = ["WS_CPU", "WS_DISCO", "AS_CPU", "AS_DISCO", "DS_CPU", "DS_DISCO"]
R = ["h", "s", "g", "v", "c", "b"]
```

Matriz de demandas

La matriz de demandas que contiene las demandas de cada recurso por cada clase de operación también ha sido extraída del libro de Menascé.

Table 8.4. Service Demands (in sec) for Auction Site Queuing Model

Device	(h)	(s)	(v)	(g)	(c)	(b)
WS-CPU	0.008	0.009	0.011	0.060	0.012	0.015
WS-disk	0.030	0.010	0.010	0.010	0.010	0.010
AS-CPU	0.000	0.030	0.035	0.025	0.045	0.040
AS-disk	0.000	0.008	0.080	0.009	0.011	0.012
DS-CPU	0.000	0.010	0.009	0.015	0.070	0.045
DS-disk	0.000	0.035	0.018	0.050	0.080	0.090

A continuación, se puede observar que hemos creado una matriz llamda `D`, la cual contiene la matriz de demandas mencionada.

In [6]:
```
D = np.array(
    [
        [0.008, 0.009, 0.011, 0.060, 0.012, 0.015],
        [0.030, 0.010, 0.010, 0.010, 0.010, 0.010],
        [0.000, 0.030, 0.035, 0.025, 0.045, 0.040],
        [0.000, 0.008, 0.080, 0.009, 0.011, 0.012],
        [0.000, 0.010, 0.009, 0.015, 0.070, 0.045],
        [0.000, 0.035, 0.018, 0.050, 0.080, 0.090],
```

```
    ]
)
```

Configuración del servicio

Para llevar a cabo este trabajo se emplea la configuración básica. Dicha configuración consiste en tener una única instancia de cada servidor y se representa de la siguiente manera:

$$< N_{WS}, N_{AS}, N_{DS} >=< 1, 1, 1 >$$

Sin embargo, para que en este trabajo podamos emplear una notación más compacta emplenado el índice $i \in K$, lo que se hace es consisderar el siguiente vector de configuración $\vec{N} =< N_i >$ con $i \in K$. En nuestro trabajo, dicho vector será un vector de todo unos.

A continuación, se puede ver que hemos creado una variable llamada `N`, la cual representa el vector de configuarción $\vec{N}$.

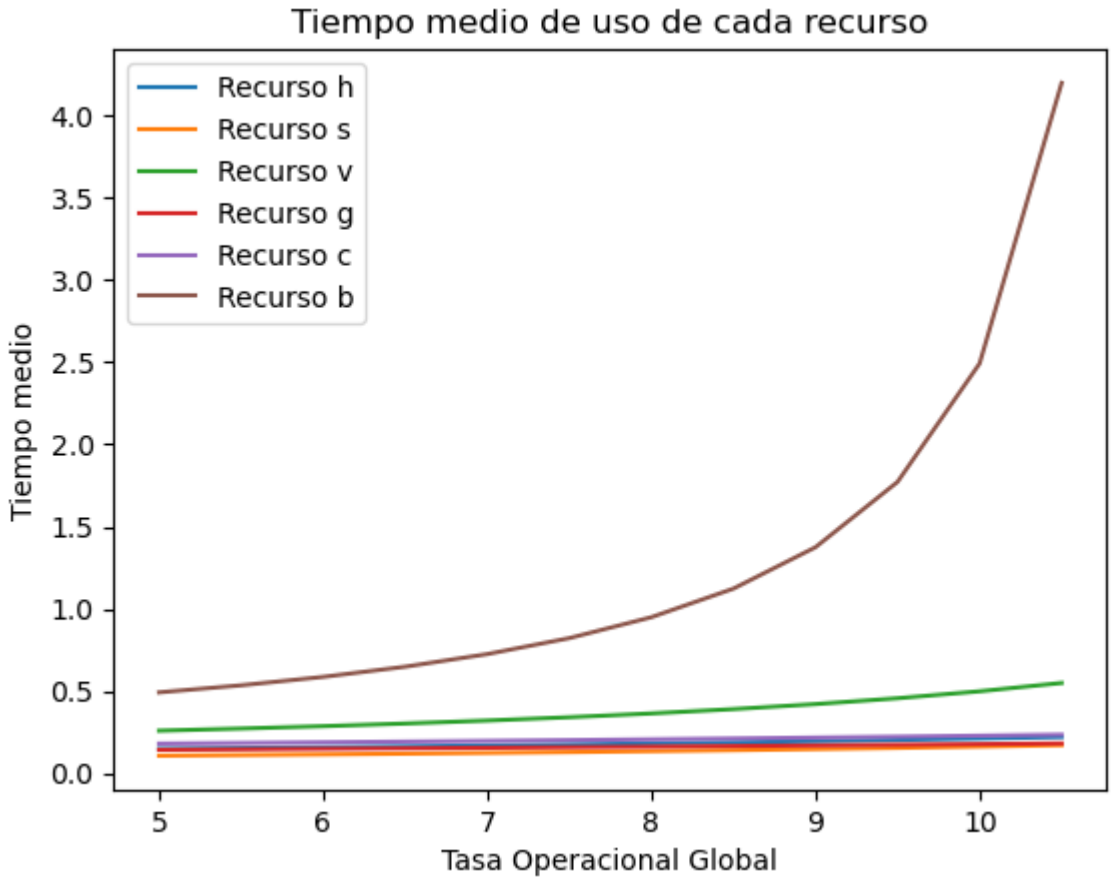
In [7]:
```
N = np.ones(len(K), dtype=int)
```

Cuestiones

1. A partir de la configuración base, representar gráficamente $T_r(\lambda)$, para cada clase de operación r ∈ R, variando λ desde 5 op/sg a intervalos de 0,5 op/sg, hasta que algún recurso i ∈ K presente una utilización $U_i \approx 0,90$.

In [8]:
```
def calcular_tr_base_tasa_global(D, N, v_a, v_b, f_a, f_b):
    tasa_op_global = 4.5
    u_max = 0
    tasas_list = []
    T_r = []
    while u_max < 0.9:
        tasa_op_global += 0.5
        tasas = utils.calcular_tasa_operaciones_por_estado(
            tasa_op_global, v_a, v_b, f_a, f_b
        )
        u = [0] * len(tasas)  # utilizacion de cada recurso
        for i in range(len(N)):
            u[i] = utils.calcular_utilizacion_recurso(tasas, D[i, :], N[i])
        u_max = max(u)
        tasas_list.append(tasa_op_global)

        # Calcular tiempo de respuesta para cada operación
        t_r = utils.calcular_tiempo_respuesta_por_operacion(D, tasas, N)
        T_r.append(t_r)
    return tasas_list, T_r


tasas_list, T_r = calcular_tr_base_tasa_global(D, N, v_a, v_b, f_a, f_b)
```

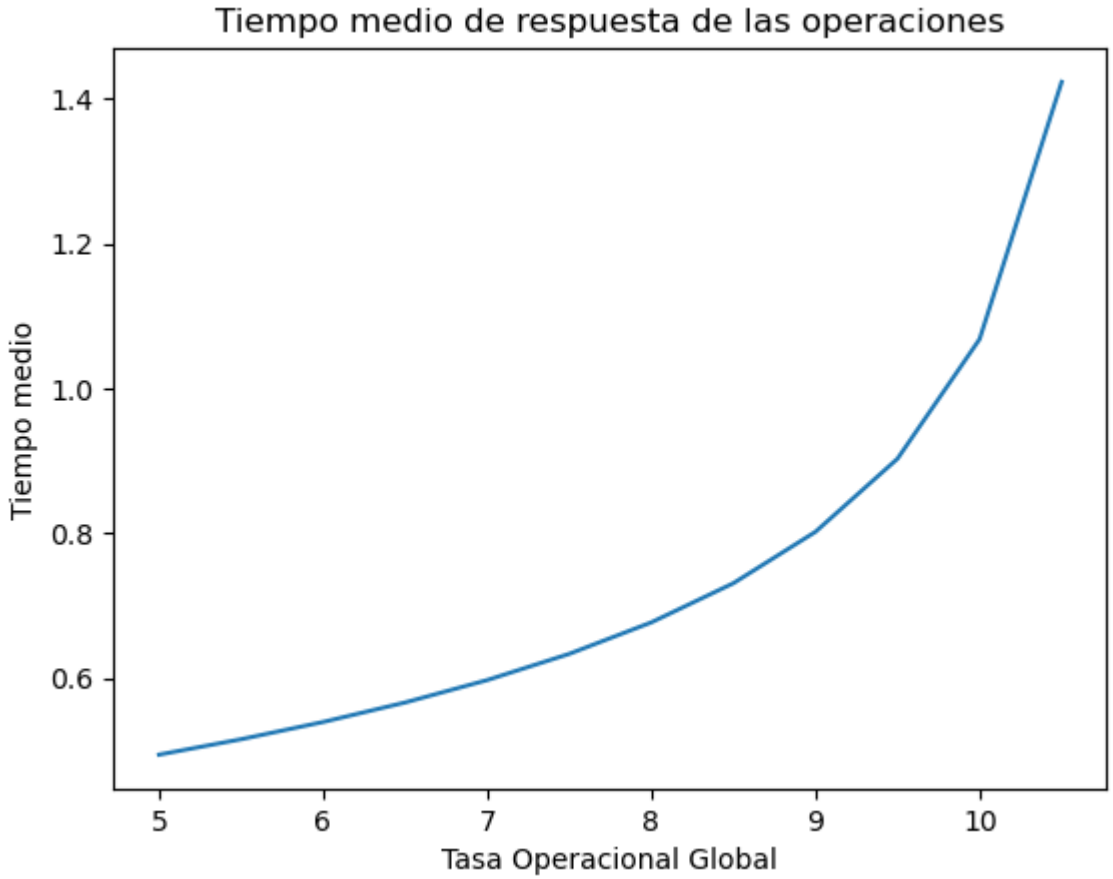
```
# Tiempo de respuesta para cada operacion (Tr)
# Array con los nombres recursos: e, h, s, v, g, c, b, x (x es el estado de sali
R = ["h", "s", "v", "g", "c", "b"]
for i in range(len(R)):
    plt.plot(tasas_list, [T_r[j][i] for j in range(len(T_r))], label=f"Recurso {

plt.title("Tiempo medio de uso de cada recurso")
plt.xlabel("Tasa Operacional Global")
plt.ylabel("Tiempo medio")
plt.legend()
plt.show()
```



2. Sobre la configuración base, represetar T(λ) siguiendo la experimentación anterior. Se establece en el SLA del servicio un tiempo T_{SLA} de como mucho 4 sg. Indica si este valor es razonable.

In [9]:
```
def calcular_t_base_tasa_global(D, N, v_a, v_b, f_a, f_b):
    tasa_op_global = 4.5
    u_max = 0
    tasas_list = []
    tiempo_medio_res = []
    while u_max < 0.9:
        tasa_op_global += 0.5
        tasas = utils.calcular_tasa_operaciones_por_estado(
            tasa_op_global, v_a, v_b, f_a, f_b
        )
        u = [0] * len(tasas)  # utilizacion de cada recurso
        for i in range(len(N)):
            u[i] = utils.calcular_utilizacion_recurso(tasas, D[i, :], N[i])
        u_max = max(u)
```

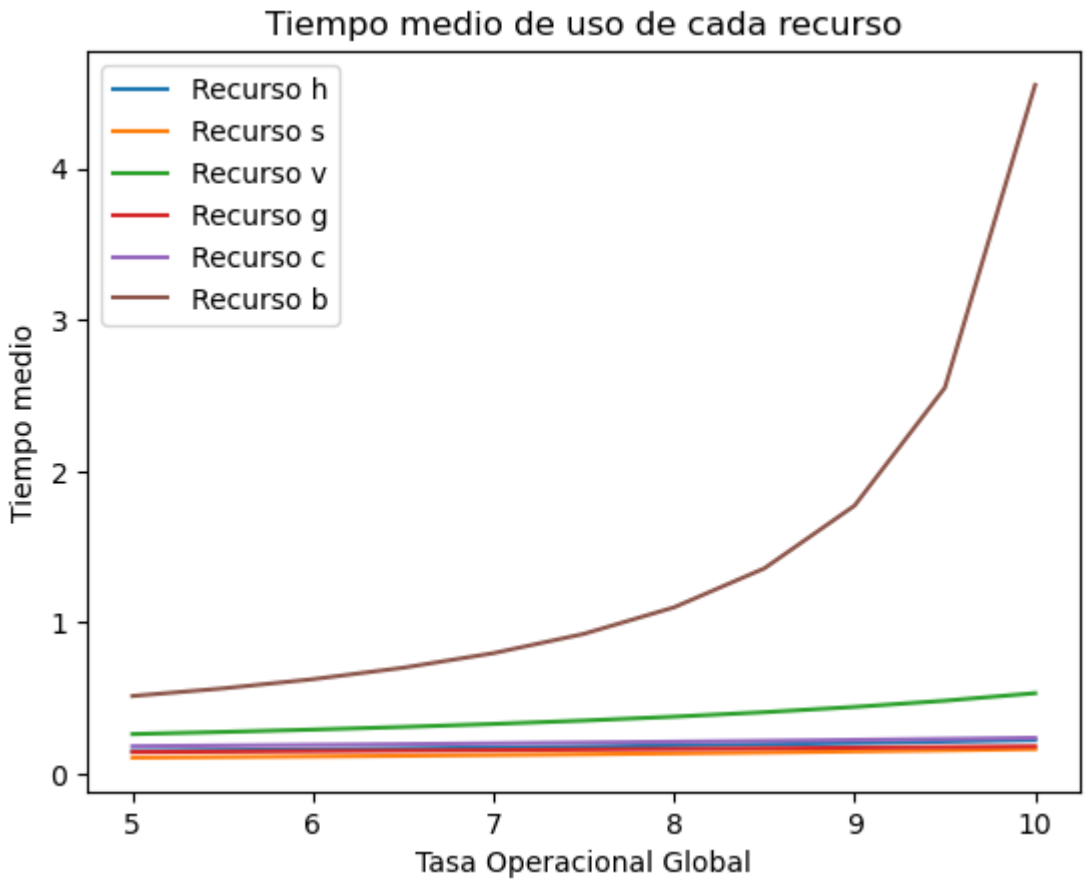
```
        tasas_list.append(tasa_op_global)

        # Calcular tiempo de residencia de cada estado
        t_r = utils.calcular_tiempo_respuesta_por_operacion(D, tasas, N)
        # Calcular tiempo medio de respuesta de las operaciones
        tiempo_medio_res.append(
            utils.calcular_tiempo_medio_respuesta_operaciones(
                tasas, tasa_op_global, t_r
            )
        )
    return tasas_list, tiempo_medio_res


tasas_list, tiempo_medio_res = calcular_t_base_tasa_global(D, N, v_a, v_b, f_a,

# T
plt.plot(tasas_list, tiempo_medio_res)
plt.title("Tiempo medio de respuesta de las operaciones")
plt.xlabel("Tasa Operacional Global")
plt.ylabel("Tiempo medio")
plt.show()
```

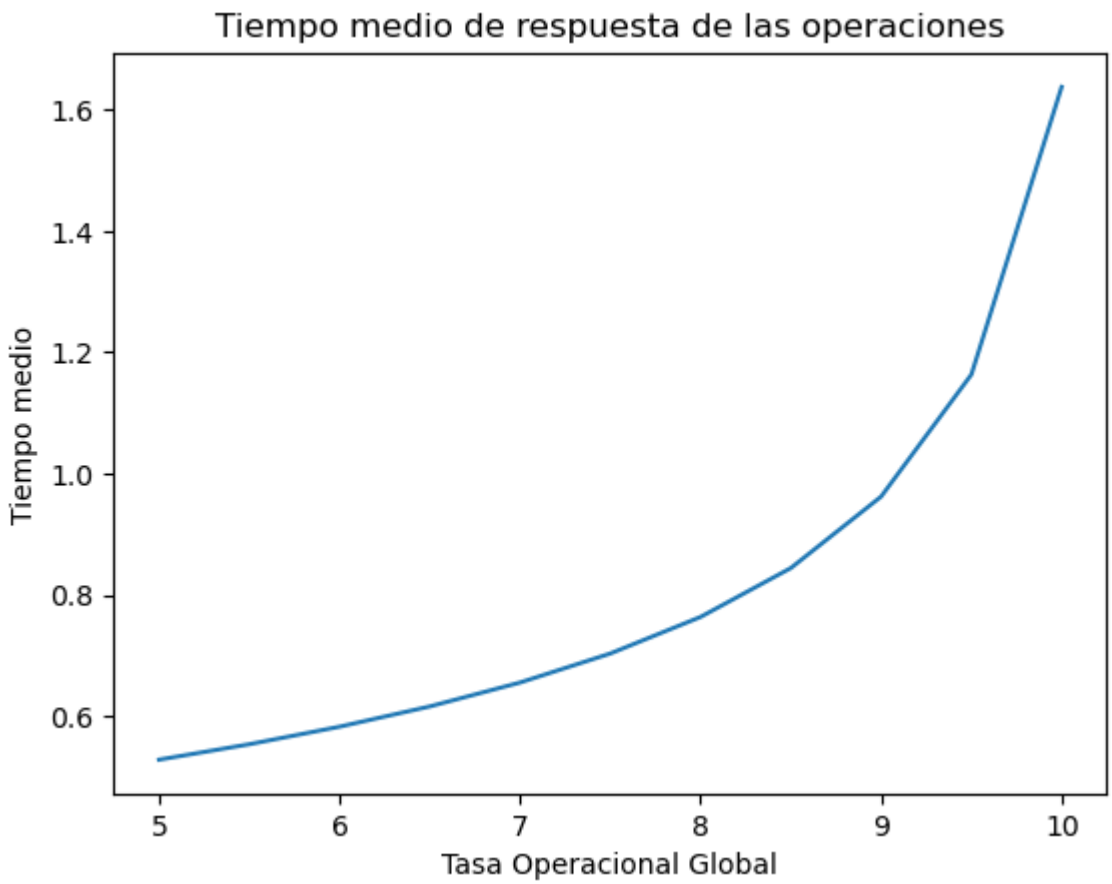


Como se puede observar en el gráfico anterior, el máximo tiempo de respuesta de las operaciones que hemos obtenido es 1,1sg, por lo que el valor máximo para T establecido en el SLA ($T_{SLA} = 4sg$) es bastante razonable.

3. Considera que la mezcla de tipos de usuario es fA = 0, 6 y fB = 0, 4. Realiza la misma experimentación anterior.

In [10]:
```
f_a = 0.6
f_b = 0.4


def calcular_tr_t_base_tasa_global(D, N, v_a, v_b, f_a, f_b):
    tasa_op_global = 4.5
    u_max = 0
    tasas_list = []
    u_list = []
    T_r = []
    tiempo_medio_res = []
    while u_max < 0.9:
        tasa_op_global += 0.5
        tasas = utils.calcular_tasa_operaciones_por_estado(
            tasa_op_global, v_a, v_b, f_a, f_b
        )
        u = [0] * len(tasas)  # utilizacion de cada recurso
        for i in range(len(N)):
            u[i] = utils.calcular_utilizacion_recurso(tasas, D[i, :], N[i])
        u_max = max(u)
        tasas_list.append(tasa_op_global)
        u_list.append(u)

        # Calcular tiempo de residencia de cada estado
        t_r = utils.calcular_tiempo_respuesta_por_operacion(D, tasas, N)
        T_r.append(t_r)
        # Calcular tiempo medio de respuesta de las operaciones
        tiempo_medio_res.append(
            utils.calcular_tiempo_medio_respuesta_operaciones(
                tasas, tasa_op_global, t_r
            )
        )
    return tasas_list, T_r, tiempo_medio_res


tasas_list, T_r, tiempo_medio_res = calcular_tr_t_base_tasa_global(
    D, N, v_a, v_b, f_a, f_b
)

# Tiempo de respuesta para cada operacion (Tr)
# Array con los nombres recursos: e, h, s, v, g, c, b, x (x es el estado de sali
R = ["h", "s", "v", "g", "c", "b"]
for i in range(len(R)):
    plt.plot(tasas_list, [T_r[j][i] for j in range(len(T_r))], label=f"Recurso {


plt.title("Tiempo medio de uso de cada recurso")
plt.xlabel("Tasa Operacional Global")
plt.ylabel("Tiempo medio")
plt.legend()
plt.show()

# T
plt.plot(tasas_list, tiempo_medio_res)
plt.title("Tiempo medio de respuesta de las operaciones")
plt.xlabel("Tasa Operacional Global")
plt.ylabel("Tiempo medio")
plt.show()
```





4. Para la configuración base y los mismos tipos de usuario calcula de manera teórica la máxima carga soportable por el sistema λ_{max} en función de la fracción de los distintos tipos

de usuarios, de las demandas de los recursos, y de la fracción de visitas a cada estado del diagrama funcional.

Sabemos que la configuración base viene dada por el vector $\vec{N}$, el cual es, como hemos comentado previamente, un vector de todo unos. Los tipos de usuarios empleados son A y B, es decir, $\alpha \in A, B$. Además, cabe decir que, para esta explicación, emplearemos $i \in K$, donde K es el vector de recursos previamente visto, y también emplearemos $r \in R$, donde R es el vector de las diferentes clases de operaciones que consumen recursos de computación, como hemos visto previamente.

Se nos pide calcular λ_{max} en función de la fracción de los distintos tipos de usuarios (f_A y f_B), de las demandas de los recursos (D), y de la fracción de visitas a cada estado del diagrama funcional ($v^{\alpha}[r]$).

Sabemos que cuando la carga global del sistema es máxima, es decir, cuando se produce λ_{max}, entonces la utilización de cada uno de los recursos del sistema es la máxima, es decir, $U_i = 1$.

Teniendo en cuenta todo lo anterior, partamos de la ecuación de utilización de un recurso para todas las clases de operaciones:

$$U_i = \sum_{r \in R} \frac{\lambda_r}{N_i} * D_{i,r}$$

Sustituyamos en la ecuación U_i por 1 y N_i por 1.

$$1 = \sum_{r \in R} \lambda_r * D_{i,r}$$

Además, sabemos que λ_r se calcula, para nuestro caso que tenemos usuarios de tipo A y de tipo B, de la siguiente manera:

$$\lambda_r = \lambda * (f_A * v^A[r] + f_B * v^B[r])$$

En este caso, podemos sustuir λ_r en la ecuación de U_i, pero teniendo en cuenta que en este caso λ será máxima para el recurso i-ésimo, es decir, será $\lambda_{max}(i)$.

$$1 = \sum_{r \in R} (\lambda_{max}(i) * (f_A * v^A[r] + f_B * v^B[r])) * D_{i,r}$$

Ahora, despejamos $\lambda_{max}(i)$.

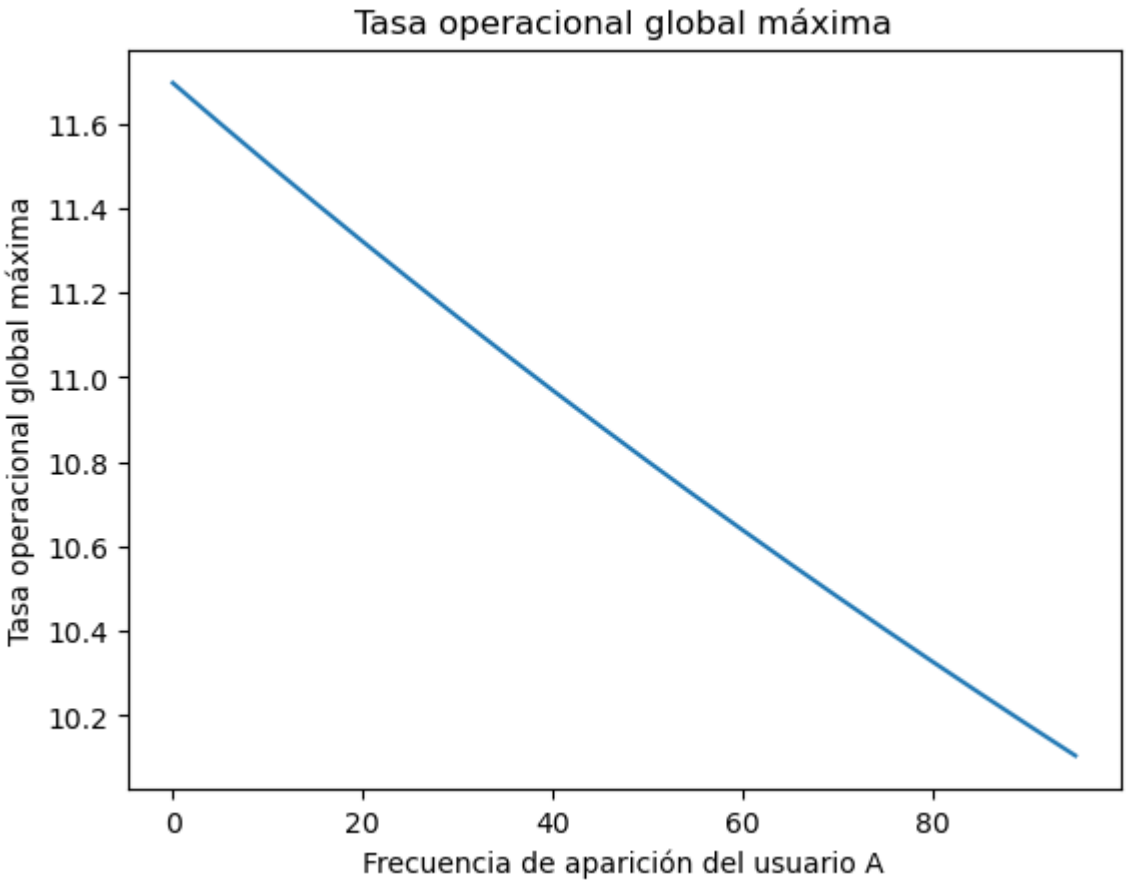
$$\lambda_{max}(i) = \frac{1}{\sum_{r \in R} (f_A * v^A[r] + f_B * v^B[r]) * D_{i,r}}$$

Una vez que ya hemos obtenido la ecuación para $\lambda_{max}(i)$, lo que tenemos que hacer es obtener la ecuación para λ_{max}. La carga de operaciones global máxima soportable por el sistema será la mínima entre las cargas máximas que han sido obtenidas para cada uno de los recursos. Por tanto, la ecuación de λ_{max} es la siguiente:

$$\lambda_{max} = \min(\lambda_{max}(i) : i \in K)$$

5. Representa la gráfica $\lambda_{max}(f_A)$ en función de la fracción f_A. Indica cuál es la mezcla (f'_A, f'_B) de tipos de usuario más favorable desde el punto de vista de λ_{max}.

In [11]:
```
tasa_op_global_max = []
f_a = 0
f_b = 1 - f_a
while f_a <= 1:
    tasa_op_global_maximos_i = []
    for i in range(len(N)):
        tasa_op_global_maximos_i.append(1 / (np.sum((v_a * f_a + v_b * f_b) * D[
    tasa_op_global_max.append(min(tasa_op_global_maximos_i))
    f_a += 0.05
    f_b = 1 - f_a

#Tasa operacional global máxima (λmax)
plt.plot(
    [i for i in range(0, 100, 5)],
    [tasa_op_global_max[j] for j in range(len(tasa_op_global_max))]
)

plt.title("Tasa operacional global máxima")
plt.xlabel("Frecuencia de aparición del usuario A")
plt.ylabel("Tasa operacional global máxima")
plt.show()
```



Como se puede observar en la gráfica anterior, la mezcla (f'_A, f'_B) de tipos se usuario más favorable desde el pinto de vista de λ_{max} es aquella en la $f'_A = 0$ y $f'_B = 1$, es

decir, el caso en el que la frecuencia de aparición del usuario A es 0 y la del usuario B es 1, siendo este el caso en el que únicamente hay usuarios de tipo B.

Cuestiones multiservidor

Asumiendo la mezcla de usarios más favorable para el sistema, calcula el tiempo medio de respuesta $T(\vec{N}, \lambda)$ para diferentes configuraciones y cargas globales de entrada. Considera que el número máximo de servidores en total no supera a 20 servidores, $\sum_{i \in K} N_i \leq 16$. Guarda los resultados discretos $T(\gamma) = T(\vec{N}(\gamma), \lambda(\gamma))$ para formar una base de datos de aprendizaje. γ indica el número de experimento realizado. Muestra una tabla con la configuración, la carga máxima soportada hasta que algún recurso alcance una utilización $U_i = 0,9$ y el tiempo de respuesta de dicha carga. Compara los resultados con la configuración base y entre diferentes configuraciones.

Suponiendo que la mezcla de usuarios más favorable para el sistema es la que se puede ver en la gráfica anterior, es decir, $f_A = 0 y f_B = 1$, vamos a calcular el tiempo medio de respuesta $T(\vec{N}, \lambda)$ para diferentes configuraciones y cargas globales de entrada. Para ello, vamos a emplear la siguiente ecuación:

$$T_{i,r} = D_{i,r} * \frac{N_i - 1}{N_i} + \frac{D_{i,r}}{1 - \sum_{r \in R} \frac{\lambda_r}{N_i} * D_{i,r}}$$

En ella se añade una penalización según el criterio de Seidmanns.

In [12]:
```
def calcular_t_base_tasa_global(D, N, v_a, v_b, f_a, f_b):
    tasa_op_global = 4.5
    u_max = 0
    tiempo_medio_res = np.inf
    while u_max < 0.9:
        tasa_op_global += 0.5
        tasas = utils.calcular_tasa_operaciones_por_estado(
            tasa_op_global, v_a, v_b, f_a, f_b
        )
        u = [0] * len(tasas)  # utilizacion de cada recurso
        for i in range(len(N)):
            u[i] = utils.calcular_utilizacion_recurso(tasas, D[i, :], N[i])
        u_max = max(u)
        # Calcular tiempo de residencia de cada estado
        t_r = utils.calcular_tiempo_respuesta_por_operacion(D, tasas, N)
        # Calcular tiempo medio de respuesta de las operaciones redondeado a 4 d
        tiempo_medio_res = utils.calcular_tiempo_medio_respuesta_operaciones(
            tasas, tasa_op_global, t_r
        ).round(4)
    return tasa_op_global, tiempo_medio_res
```

```python
def generar_combinaciones_N():
    # Define the maximum value for each N_i and the maximum sum for N
    max_value = 16
    max_sum = 20
    # Generate all combinations of N
    return (
        (x, x, y, y, z, z)
        for x in range(1, max_value + 1)
        for y in range(1, max_value + 1)
        for z in range(1, max_value + 1)
        if x + y + z < max_sum
    )


base_datos_aprendizaje = []

combinaciones = generar_combinaciones_N()
try:
    # Es un bucle infinito, por que no hay una forma de saber cuando se acaban l
    # hasta que se llega a la última y se lanza una excepción tipo StopIteration
    while True:
        N = next(combinaciones)
        f_a = 0
        f_b = 1 - f_a
        tasa_global, tiempo_medio_res = calcular_t_base_tasa_global(
            D, N, v_a, v_b, f_a, f_b
        )
        base_datos_aprendizaje.append((N, tasa_global, tiempo_medio_res))
except StopIteration:
    print("No hay más combinaciones posibles")

# Mostrar los resultados de la base de datos de aprendizaje de forma ordenada y
print("|N|Tasa Global(op/s)|Tiempo medio de respuesta(s)|")
for i in range(len(base_datos_aprendizaje)):
    print(
        f"|{base_datos_aprendizaje[i][0]}|{base_datos_aprendizaje[i][1]}|{base_d
    )

# Guardar la base de datos de aprendizaje en un fichero
with open("base_datos_aprendizaje.txt", "w") as f:
    for i in range(len(base_datos_aprendizaje)):
        f.write(
            f"{base_datos_aprendizaje[i][0]} {base_datos_aprendizaje[i][1]} {bas
        )
```

```
No hay más combinaciones posibles
|N|Tasa Global(op/s)|Tiempo medio de respuesta(s)|
|(1, 1, 1, 1, 1, 1)|11.0|1.4011|
|(1, 1, 1, 1, 2, 2)|14.5|1.2933|
|(1, 1, 1, 1, 3, 3)|14.5|1.2494|
|(1, 1, 1, 1, 4, 4)|14.5|1.237|
|(1, 1, 1, 1, 5, 5)|14.5|1.2312|
|(1, 1, 1, 1, 6, 6)|14.5|1.2278|
|(1, 1, 1, 1, 7, 7)|14.5|1.2255|
|(1, 1, 1, 1, 8, 8)|14.5|1.224|
|(1, 1, 1, 1, 9, 9)|14.5|1.2228|
|(1, 1, 1, 1, 10, 10)|14.5|1.2219|
|(1, 1, 1, 1, 11, 11)|14.5|1.2211|
|(1, 1, 1, 1, 12, 12)|14.5|1.2205|
|(1, 1, 1, 1, 13, 13)|14.5|1.2201|
|(1, 1, 1, 1, 14, 14)|14.5|1.2196|
|(1, 1, 1, 1, 15, 15)|14.5|1.2193|
|(1, 1, 1, 1, 16, 16)|14.5|1.219|
|(1, 1, 2, 2, 1, 1)|11.0|1.331|
|(1, 1, 2, 2, 2, 2)|18.0|1.9392|
|(1, 1, 2, 2, 3, 3)|18.0|1.8326|
|(1, 1, 2, 2, 4, 4)|18.0|1.8121|
|(1, 1, 2, 2, 5, 5)|18.0|1.8033|
|(1, 1, 2, 2, 6, 6)|18.0|1.7985|
|(1, 1, 2, 2, 7, 7)|18.0|1.7954|
|(1, 1, 2, 2, 8, 8)|18.0|1.7932|
|(1, 1, 2, 2, 9, 9)|18.0|1.7916|
|(1, 1, 2, 2, 10, 10)|18.0|1.7904|
|(1, 1, 2, 2, 11, 11)|18.0|1.7894|
|(1, 1, 2, 2, 12, 12)|18.0|1.7887|
|(1, 1, 2, 2, 13, 13)|18.0|1.788|
|(1, 1, 2, 2, 14, 14)|18.0|1.7875|
|(1, 1, 2, 2, 15, 15)|18.0|1.787|
|(1, 1, 2, 2, 16, 16)|18.0|1.7866|
|(1, 1, 3, 3, 1, 1)|11.0|1.3204|
|(1, 1, 3, 3, 2, 2)|18.0|1.9096|
|(1, 1, 3, 3, 3, 3)|18.0|1.803|
|(1, 1, 3, 3, 4, 4)|18.0|1.7825|
|(1, 1, 3, 3, 5, 5)|18.0|1.7738|
|(1, 1, 3, 3, 6, 6)|18.0|1.7689|
|(1, 1, 3, 3, 7, 7)|18.0|1.7658|
|(1, 1, 3, 3, 8, 8)|18.0|1.7636|
|(1, 1, 3, 3, 9, 9)|18.0|1.762|
|(1, 1, 3, 3, 10, 10)|18.0|1.7608|
|(1, 1, 3, 3, 11, 11)|18.0|1.7599|
|(1, 1, 3, 3, 12, 12)|18.0|1.7591|
|(1, 1, 3, 3, 13, 13)|18.0|1.7584|
|(1, 1, 3, 3, 14, 14)|18.0|1.7579|
|(1, 1, 3, 3, 15, 15)|18.0|1.7575|
|(1, 1, 4, 4, 1, 1)|11.0|1.3161|
|(1, 1, 4, 4, 2, 2)|18.0|1.9|
|(1, 1, 4, 4, 3, 3)|18.0|1.7935|
|(1, 1, 4, 4, 4, 4)|18.0|1.773|
|(1, 1, 4, 4, 5, 5)|18.0|1.7642|
|(1, 1, 4, 4, 6, 6)|18.0|1.7593|
|(1, 1, 4, 4, 7, 7)|18.0|1.7562|
|(1, 1, 4, 4, 8, 8)|18.0|1.754|
|(1, 1, 4, 4, 9, 9)|18.0|1.7525|
|(1, 1, 4, 4, 10, 10)|18.0|1.7512|
|(1, 1, 4, 4, 11, 11)|18.0|1.7503|
```

```
|(1, 1, 4, 4, 12, 12)|18.0|1.7495|
|(1, 1, 4, 4, 13, 13)|18.0|1.7489|
|(1, 1, 4, 4, 14, 14)|18.0|1.7483|
|(1, 1, 5, 5, 1, 1)|11.0|1.3137|
|(1, 1, 5, 5, 2, 2)|18.0|1.8952|
|(1, 1, 5, 5, 3, 3)|18.0|1.7887|
|(1, 1, 5, 5, 4, 4)|18.0|1.7682|
|(1, 1, 5, 5, 5, 5)|18.0|1.7594|
|(1, 1, 5, 5, 6, 6)|18.0|1.7545|
|(1, 1, 5, 5, 7, 7)|18.0|1.7514|
|(1, 1, 5, 5, 8, 8)|18.0|1.7492|
|(1, 1, 5, 5, 9, 9)|18.0|1.7477|
|(1, 1, 5, 5, 10, 10)|18.0|1.7465|
|(1, 1, 5, 5, 11, 11)|18.0|1.7455|
|(1, 1, 5, 5, 12, 12)|18.0|1.7447|
|(1, 1, 5, 5, 13, 13)|18.0|1.7441|
|(1, 1, 6, 6, 1, 1)|11.0|1.3122|
|(1, 1, 6, 6, 2, 2)|18.0|1.8924|
|(1, 1, 6, 6, 3, 3)|18.0|1.7858|
|(1, 1, 6, 6, 4, 4)|18.0|1.7653|
|(1, 1, 6, 6, 5, 5)|18.0|1.7565|
|(1, 1, 6, 6, 6, 6)|18.0|1.7516|
|(1, 1, 6, 6, 7, 7)|18.0|1.7485|
|(1, 1, 6, 6, 8, 8)|18.0|1.7464|
|(1, 1, 6, 6, 9, 9)|18.0|1.7448|
|(1, 1, 6, 6, 10, 10)|18.0|1.7436|
|(1, 1, 6, 6, 11, 11)|18.0|1.7426|
|(1, 1, 6, 6, 12, 12)|18.0|1.7418|
|(1, 1, 7, 7, 1, 1)|11.0|1.3112|
|(1, 1, 7, 7, 2, 2)|18.0|1.8904|
|(1, 1, 7, 7, 3, 3)|18.0|1.7839|
|(1, 1, 7, 7, 4, 4)|18.0|1.7634|
|(1, 1, 7, 7, 5, 5)|18.0|1.7546|
|(1, 1, 7, 7, 6, 6)|18.0|1.7497|
|(1, 1, 7, 7, 7, 7)|18.0|1.7466|
|(1, 1, 7, 7, 8, 8)|18.0|1.7445|
|(1, 1, 7, 7, 9, 9)|18.0|1.7429|
|(1, 1, 7, 7, 10, 10)|18.0|1.7417|
|(1, 1, 7, 7, 11, 11)|18.0|1.7407|
|(1, 1, 8, 8, 1, 1)|11.0|1.3104|
|(1, 1, 8, 8, 2, 2)|18.0|1.8891|
|(1, 1, 8, 8, 3, 3)|18.0|1.7825|
|(1, 1, 8, 8, 4, 4)|18.0|1.762|
|(1, 1, 8, 8, 5, 5)|18.0|1.7532|
|(1, 1, 8, 8, 6, 6)|18.0|1.7483|
|(1, 1, 8, 8, 7, 7)|18.0|1.7452|
|(1, 1, 8, 8, 8, 8)|18.0|1.7431|
|(1, 1, 8, 8, 9, 9)|18.0|1.7415|
|(1, 1, 8, 8, 10, 10)|18.0|1.7403|
|(1, 1, 9, 9, 1, 1)|11.0|1.3098|
|(1, 1, 9, 9, 2, 2)|18.0|1.888|
|(1, 1, 9, 9, 3, 3)|18.0|1.7815|
|(1, 1, 9, 9, 4, 4)|18.0|1.761|
|(1, 1, 9, 9, 5, 5)|18.0|1.7522|
|(1, 1, 9, 9, 6, 6)|18.0|1.7473|
|(1, 1, 9, 9, 7, 7)|18.0|1.7442|
|(1, 1, 9, 9, 8, 8)|18.0|1.742|
|(1, 1, 9, 9, 9, 9)|18.0|1.7405|
|(1, 1, 10, 10, 1, 1)|11.0|1.3094|
|(1, 1, 10, 10, 2, 2)|18.0|1.8872|
```

```
|(1, 1, 10, 10, 3, 3)|18.0|1.7807|
|(1, 1, 10, 10, 4, 4)|18.0|1.7602|
|(1, 1, 10, 10, 5, 5)|18.0|1.7514|
|(1, 1, 10, 10, 6, 6)|18.0|1.7465|
|(1, 1, 10, 10, 7, 7)|18.0|1.7434|
|(1, 1, 10, 10, 8, 8)|18.0|1.7412|
|(1, 1, 11, 11, 1, 1)|11.0|1.309|
|(1, 1, 11, 11, 2, 2)|18.0|1.8866|
|(1, 1, 11, 11, 3, 3)|18.0|1.78|
|(1, 1, 11, 11, 4, 4)|18.0|1.7595|
|(1, 1, 11, 11, 5, 5)|18.0|1.7508|
|(1, 1, 11, 11, 6, 6)|18.0|1.7459|
|(1, 1, 11, 11, 7, 7)|18.0|1.7428|
|(1, 1, 12, 12, 1, 1)|11.0|1.3087|
|(1, 1, 12, 12, 2, 2)|18.0|1.8861|
|(1, 1, 12, 12, 3, 3)|18.0|1.7795|
|(1, 1, 12, 12, 4, 4)|18.0|1.759|
|(1, 1, 12, 12, 5, 5)|18.0|1.7502|
|(1, 1, 12, 12, 6, 6)|18.0|1.7453|
|(1, 1, 13, 13, 1, 1)|11.0|1.3085|
|(1, 1, 13, 13, 2, 2)|18.0|1.8856|
|(1, 1, 13, 13, 3, 3)|18.0|1.7791|
|(1, 1, 13, 13, 4, 4)|18.0|1.7586|
|(1, 1, 13, 13, 5, 5)|18.0|1.7498|
|(1, 1, 14, 14, 1, 1)|11.0|1.3083|
|(1, 1, 14, 14, 2, 2)|18.0|1.8852|
|(1, 1, 14, 14, 3, 3)|18.0|1.7787|
|(1, 1, 14, 14, 4, 4)|18.0|1.7582|
|(1, 1, 15, 15, 1, 1)|11.0|1.3081|
|(1, 1, 15, 15, 2, 2)|18.0|1.8849|
|(1, 1, 15, 15, 3, 3)|18.0|1.7784|
|(1, 1, 16, 16, 1, 1)|11.0|1.3079|
|(1, 1, 16, 16, 2, 2)|18.0|1.8846|
|(2, 2, 1, 1, 1, 1)|11.0|1.2451|
|(2, 2, 1, 1, 2, 2)|14.5|0.9357|
|(2, 2, 1, 1, 3, 3)|14.5|0.8918|
|(2, 2, 1, 1, 4, 4)|14.5|0.8794|
|(2, 2, 1, 1, 5, 5)|14.5|0.8736|
|(2, 2, 1, 1, 6, 6)|14.5|0.8702|
|(2, 2, 1, 1, 7, 7)|14.5|0.8679|
|(2, 2, 1, 1, 8, 8)|14.5|0.8663|
|(2, 2, 1, 1, 9, 9)|14.5|0.8651|
|(2, 2, 1, 1, 10, 10)|14.5|0.8642|
|(2, 2, 1, 1, 11, 11)|14.5|0.8635|
|(2, 2, 1, 1, 12, 12)|14.5|0.8629|
|(2, 2, 1, 1, 13, 13)|14.5|0.8624|
|(2, 2, 1, 1, 14, 14)|14.5|0.862|
|(2, 2, 1, 1, 15, 15)|14.5|0.8617|
|(2, 2, 1, 1, 16, 16)|14.5|0.8614|
|(2, 2, 2, 2, 1, 1)|11.0|1.175|
|(2, 2, 2, 2, 2, 2)|21.5|1.1813|
|(2, 2, 2, 2, 3, 3)|28.5|1.3272|
|(2, 2, 2, 2, 4, 4)|28.5|1.199|
|(2, 2, 2, 2, 5, 5)|28.5|1.1702|
|(2, 2, 2, 2, 6, 6)|28.5|1.1574|
|(2, 2, 2, 2, 7, 7)|28.5|1.1502|
|(2, 2, 2, 2, 8, 8)|28.5|1.1455|
|(2, 2, 2, 2, 9, 9)|28.5|1.1422|
|(2, 2, 2, 2, 10, 10)|28.5|1.1398|
|(2, 2, 2, 2, 11, 11)|28.5|1.138|
```

```
|(2, 2, 2, 2, 12, 12)|28.5|1.1365|
|(2, 2, 2, 2, 13, 13)|28.5|1.1353|
|(2, 2, 2, 2, 14, 14)|28.5|1.1343|
|(2, 2, 2, 2, 15, 15)|28.5|1.1335|
|(2, 2, 3, 3, 1, 1)|11.0|1.1644|
|(2, 2, 3, 3, 2, 2)|21.5|1.1301|
|(2, 2, 3, 3, 3, 3)|32.0|1.5928|
|(2, 2, 3, 3, 4, 4)|36.0|2.0079|
|(2, 2, 3, 3, 5, 5)|36.0|1.9272|
|(2, 2, 3, 3, 6, 6)|36.0|1.9013|
|(2, 2, 3, 3, 7, 7)|36.0|1.8885|
|(2, 2, 3, 3, 8, 8)|36.0|1.8808|
|(2, 2, 3, 3, 9, 9)|36.0|1.8757|
|(2, 2, 3, 3, 10, 10)|36.0|1.872|
|(2, 2, 3, 3, 11, 11)|36.0|1.8693|
|(2, 2, 3, 3, 12, 12)|36.0|1.8672|
|(2, 2, 3, 3, 13, 13)|36.0|1.8654|
|(2, 2, 3, 3, 14, 14)|36.0|1.864|
|(2, 2, 4, 4, 1, 1)|11.0|1.1601|
|(2, 2, 4, 4, 2, 2)|21.5|1.1165|
|(2, 2, 4, 4, 3, 3)|32.0|1.5517|
|(2, 2, 4, 4, 4, 4)|36.0|1.9392|
|(2, 2, 4, 4, 5, 5)|36.0|1.8585|
|(2, 2, 4, 4, 6, 6)|36.0|1.8326|
|(2, 2, 4, 4, 7, 7)|36.0|1.8198|
|(2, 2, 4, 4, 8, 8)|36.0|1.8121|
|(2, 2, 4, 4, 9, 9)|36.0|1.807|
|(2, 2, 4, 4, 10, 10)|36.0|1.8033|
|(2, 2, 4, 4, 11, 11)|36.0|1.8006|
|(2, 2, 4, 4, 12, 12)|36.0|1.7985|
|(2, 2, 4, 4, 13, 13)|36.0|1.7968|
|(2, 2, 5, 5, 1, 1)|11.0|1.1577|
|(2, 2, 5, 5, 2, 2)|21.5|1.1101|
|(2, 2, 5, 5, 3, 3)|32.0|1.5371|
|(2, 2, 5, 5, 4, 4)|36.0|1.9193|
|(2, 2, 5, 5, 5, 5)|36.0|1.8386|
|(2, 2, 5, 5, 6, 6)|36.0|1.8127|
|(2, 2, 5, 5, 7, 7)|36.0|1.7999|
|(2, 2, 5, 5, 8, 8)|36.0|1.7922|
|(2, 2, 5, 5, 9, 9)|36.0|1.7871|
|(2, 2, 5, 5, 10, 10)|36.0|1.7835|
|(2, 2, 5, 5, 11, 11)|36.0|1.7807|
|(2, 2, 5, 5, 12, 12)|36.0|1.7786|
|(2, 2, 6, 6, 1, 1)|11.0|1.1562|
|(2, 2, 6, 6, 2, 2)|21.5|1.1063|
|(2, 2, 6, 6, 3, 3)|32.0|1.5296|
|(2, 2, 6, 6, 4, 4)|36.0|1.9096|
|(2, 2, 6, 6, 5, 5)|36.0|1.8289|
|(2, 2, 6, 6, 6, 6)|36.0|1.803|
|(2, 2, 6, 6, 7, 7)|36.0|1.7902|
|(2, 2, 6, 6, 8, 8)|36.0|1.7825|
|(2, 2, 6, 6, 9, 9)|36.0|1.7774|
|(2, 2, 6, 6, 10, 10)|36.0|1.7738|
|(2, 2, 6, 6, 11, 11)|36.0|1.771|
|(2, 2, 7, 7, 1, 1)|11.0|1.1552|
|(2, 2, 7, 7, 2, 2)|21.5|1.1038|
|(2, 2, 7, 7, 3, 3)|32.0|1.525|
|(2, 2, 7, 7, 4, 4)|36.0|1.9038|
|(2, 2, 7, 7, 5, 5)|36.0|1.8231|
|(2, 2, 7, 7, 6, 6)|36.0|1.7973|
```

```
|(2, 2, 7, 7, 7, 7)|36.0|1.7845|
|(2, 2, 7, 7, 8, 8)|36.0|1.7768|
|(2, 2, 7, 7, 9, 9)|36.0|1.7717|
|(2, 2, 7, 7, 10, 10)|36.0|1.768|
|(2, 2, 8, 8, 1, 1)|11.0|1.1544|
|(2, 2, 8, 8, 2, 2)|21.5|1.1021|
|(2, 2, 8, 8, 3, 3)|32.0|1.5218|
|(2, 2, 8, 8, 4, 4)|36.0|1.9|
|(2, 2, 8, 8, 5, 5)|36.0|1.8193|
|(2, 2, 8, 8, 6, 6)|36.0|1.7935|
|(2, 2, 8, 8, 7, 7)|36.0|1.7806|
|(2, 2, 8, 8, 8, 8)|36.0|1.773|
|(2, 2, 8, 8, 9, 9)|36.0|1.7678|
|(2, 2, 9, 9, 1, 1)|11.0|1.1539|
|(2, 2, 9, 9, 2, 2)|21.5|1.1008|
|(2, 2, 9, 9, 3, 3)|32.0|1.5195|
|(2, 2, 9, 9, 4, 4)|36.0|1.8973|
|(2, 2, 9, 9, 5, 5)|36.0|1.8166|
|(2, 2, 9, 9, 6, 6)|36.0|1.7907|
|(2, 2, 9, 9, 7, 7)|36.0|1.7779|
|(2, 2, 9, 9, 8, 8)|36.0|1.7702|
|(2, 2, 10, 10, 1, 1)|11.0|1.1534|
|(2, 2, 10, 10, 2, 2)|21.5|1.0998|
|(2, 2, 10, 10, 3, 3)|32.0|1.5178|
|(2, 2, 10, 10, 4, 4)|36.0|1.8952|
|(2, 2, 10, 10, 5, 5)|36.0|1.8145|
|(2, 2, 10, 10, 6, 6)|36.0|1.7887|
|(2, 2, 10, 10, 7, 7)|36.0|1.7758|
|(2, 2, 11, 11, 1, 1)|11.0|1.153|
|(2, 2, 11, 11, 2, 2)|21.5|1.099|
|(2, 2, 11, 11, 3, 3)|32.0|1.5165|
|(2, 2, 11, 11, 4, 4)|36.0|1.8936|
|(2, 2, 11, 11, 5, 5)|36.0|1.8129|
|(2, 2, 11, 11, 6, 6)|36.0|1.7871|
|(2, 2, 12, 12, 1, 1)|11.0|1.1527|
|(2, 2, 12, 12, 2, 2)|21.5|1.0983|
|(2, 2, 12, 12, 3, 3)|32.0|1.5154|
|(2, 2, 12, 12, 4, 4)|36.0|1.8924|
|(2, 2, 12, 12, 5, 5)|36.0|1.8117|
|(2, 2, 13, 13, 1, 1)|11.0|1.1525|
|(2, 2, 13, 13, 2, 2)|21.5|1.0978|
|(2, 2, 13, 13, 3, 3)|32.0|1.5145|
|(2, 2, 13, 13, 4, 4)|36.0|1.8913|
|(2, 2, 14, 14, 1, 1)|11.0|1.1523|
|(2, 2, 14, 14, 2, 2)|21.5|1.0973|
|(2, 2, 14, 14, 3, 3)|32.0|1.5137|
|(2, 2, 15, 15, 1, 1)|11.0|1.1521|
|(2, 2, 15, 15, 2, 2)|21.5|1.097|
|(2, 2, 16, 16, 1, 1)|11.0|1.1519|
|(3, 3, 1, 1, 1, 1)|11.0|1.2149|
|(3, 3, 1, 1, 2, 2)|14.5|0.8877|
|(3, 3, 1, 1, 3, 3)|14.5|0.8438|
|(3, 3, 1, 1, 4, 4)|14.5|0.8314|
|(3, 3, 1, 1, 5, 5)|14.5|0.8256|
|(3, 3, 1, 1, 6, 6)|14.5|0.8222|
|(3, 3, 1, 1, 7, 7)|14.5|0.8199|
|(3, 3, 1, 1, 8, 8)|14.5|0.8184|
|(3, 3, 1, 1, 9, 9)|14.5|0.8172|
|(3, 3, 1, 1, 10, 10)|14.5|0.8163|
|(3, 3, 1, 1, 11, 11)|14.5|0.8155|
```

```
|(3, 3, 1, 1, 12, 12)|14.5|0.8149|
|(3, 3, 1, 1, 13, 13)|14.5|0.8145|
|(3, 3, 1, 1, 14, 14)|14.5|0.814|
|(3, 3, 1, 1, 15, 15)|14.5|0.8137|
|(3, 3, 2, 2, 1, 1)|11.0|1.1448|
|(3, 3, 2, 2, 2, 2)|21.5|1.0703|
|(3, 3, 2, 2, 3, 3)|28.5|1.0575|
|(3, 3, 2, 2, 4, 4)|28.5|0.9293|
|(3, 3, 2, 2, 5, 5)|28.5|0.9006|
|(3, 3, 2, 2, 6, 6)|28.5|0.8878|
|(3, 3, 2, 2, 7, 7)|28.5|0.8805|
|(3, 3, 2, 2, 8, 8)|28.5|0.8759|
|(3, 3, 2, 2, 9, 9)|28.5|0.8726|
|(3, 3, 2, 2, 10, 10)|28.5|0.8702|
|(3, 3, 2, 2, 11, 11)|28.5|0.8683|
|(3, 3, 2, 2, 12, 12)|28.5|0.8669|
|(3, 3, 2, 2, 13, 13)|28.5|0.8657|
|(3, 3, 2, 2, 14, 14)|28.5|0.8647|
|(3, 3, 3, 3, 1, 1)|11.0|1.1342|
|(3, 3, 3, 3, 2, 2)|21.5|1.0191|
|(3, 3, 3, 3, 3, 3)|32.0|1.1302|
|(3, 3, 3, 3, 4, 4)|42.5|1.558|
|(3, 3, 3, 3, 5, 5)|43.0|1.2834|
|(3, 3, 3, 3, 6, 6)|43.0|1.2277|
|(3, 3, 3, 3, 7, 7)|43.0|1.2053|
|(3, 3, 3, 3, 8, 8)|43.0|1.1931|
|(3, 3, 3, 3, 9, 9)|43.0|1.1854|
|(3, 3, 3, 3, 10, 10)|43.0|1.1802|
|(3, 3, 3, 3, 11, 11)|43.0|1.1763|
|(3, 3, 3, 3, 12, 12)|43.0|1.1734|
|(3, 3, 3, 3, 13, 13)|43.0|1.171|
|(3, 3, 4, 4, 1, 1)|11.0|1.1299|
|(3, 3, 4, 4, 2, 2)|21.5|1.0055|
|(3, 3, 4, 4, 3, 3)|32.0|1.0891|
|(3, 3, 4, 4, 4, 4)|42.5|1.3276|
|(3, 3, 4, 4, 5, 5)|53.0|2.1777|
|(3, 3, 4, 4, 6, 6)|54.0|2.1026|
|(3, 3, 4, 4, 7, 7)|54.0|2.0377|
|(3, 3, 4, 4, 8, 8)|54.0|2.0108|
|(3, 3, 4, 4, 9, 9)|54.0|1.9961|
|(3, 3, 4, 4, 10, 10)|54.0|1.9867|
|(3, 3, 4, 4, 11, 11)|54.0|1.9803|
|(3, 3, 4, 4, 12, 12)|54.0|1.9756|
|(3, 3, 5, 5, 1, 1)|11.0|1.1275|
|(3, 3, 5, 5, 2, 2)|21.5|0.9991|
|(3, 3, 5, 5, 3, 3)|32.0|1.0745|
|(3, 3, 5, 5, 4, 4)|42.5|1.293|
|(3, 3, 5, 5, 5, 5)|53.0|2.065|
|(3, 3, 5, 5, 6, 6)|54.0|1.9714|
|(3, 3, 5, 5, 7, 7)|54.0|1.9064|
|(3, 3, 5, 5, 8, 8)|54.0|1.8795|
|(3, 3, 5, 5, 9, 9)|54.0|1.8648|
|(3, 3, 5, 5, 10, 10)|54.0|1.8555|
|(3, 3, 5, 5, 11, 11)|54.0|1.849|
|(3, 3, 6, 6, 1, 1)|11.0|1.126|
|(3, 3, 6, 6, 2, 2)|21.5|0.9953|
|(3, 3, 6, 6, 3, 3)|32.0|1.067|
|(3, 3, 6, 6, 4, 4)|42.5|1.2785|
|(3, 3, 6, 6, 5, 5)|53.0|2.0352|
|(3, 3, 6, 6, 6, 6)|54.0|1.9392|
```

```
|(3, 3, 6, 6, 7, 7)|54.0|1.8742|
|(3, 3, 6, 6, 8, 8)|54.0|1.8473|
|(3, 3, 6, 6, 9, 9)|54.0|1.8326|
|(3, 3, 6, 6, 10, 10)|54.0|1.8233|
|(3, 3, 7, 7, 1, 1)|11.0|1.125|
|(3, 3, 7, 7, 2, 2)|21.5|0.9928|
|(3, 3, 7, 7, 3, 3)|32.0|1.0624|
|(3, 3, 7, 7, 4, 4)|42.5|1.2704|
|(3, 3, 7, 7, 5, 5)|53.0|2.021|
|(3, 3, 7, 7, 6, 6)|54.0|1.9242|
|(3, 3, 7, 7, 7, 7)|54.0|1.8592|
|(3, 3, 7, 7, 8, 8)|54.0|1.8323|
|(3, 3, 7, 7, 9, 9)|54.0|1.8176|
|(3, 3, 8, 8, 1, 1)|11.0|1.1242|
|(3, 3, 8, 8, 2, 2)|21.5|0.9911|
|(3, 3, 8, 8, 3, 3)|32.0|1.0592|
|(3, 3, 8, 8, 4, 4)|42.5|1.2652|
|(3, 3, 8, 8, 5, 5)|53.0|2.0126|
|(3, 3, 8, 8, 6, 6)|54.0|1.9154|
|(3, 3, 8, 8, 7, 7)|54.0|1.8504|
|(3, 3, 8, 8, 8, 8)|54.0|1.8236|
|(3, 3, 9, 9, 1, 1)|11.0|1.1237|
|(3, 3, 9, 9, 2, 2)|21.5|0.9898|
|(3, 3, 9, 9, 3, 3)|32.0|1.0569|
|(3, 3, 9, 9, 4, 4)|42.5|1.2615|
|(3, 3, 9, 9, 5, 5)|53.0|2.007|
|(3, 3, 9, 9, 6, 6)|54.0|1.9096|
|(3, 3, 9, 9, 7, 7)|54.0|1.8446|
|(3, 3, 10, 10, 1, 1)|11.0|1.1232|
|(3, 3, 10, 10, 2, 2)|21.5|0.9888|
|(3, 3, 10, 10, 3, 3)|32.0|1.0552|
|(3, 3, 10, 10, 4, 4)|42.5|1.2589|
|(3, 3, 10, 10, 5, 5)|53.0|2.003|
|(3, 3, 10, 10, 6, 6)|54.0|1.9055|
|(3, 3, 11, 11, 1, 1)|11.0|1.1229|
|(3, 3, 11, 11, 2, 2)|21.5|0.988|
|(3, 3, 11, 11, 3, 3)|32.0|1.0539|
|(3, 3, 11, 11, 4, 4)|42.5|1.2568|
|(3, 3, 11, 11, 5, 5)|53.0|2.0001|
|(3, 3, 12, 12, 1, 1)|11.0|1.1226|
|(3, 3, 12, 12, 2, 2)|21.5|0.9873|
|(3, 3, 12, 12, 3, 3)|32.0|1.0528|
|(3, 3, 12, 12, 4, 4)|42.5|1.2552|
|(3, 3, 13, 13, 1, 1)|11.0|1.1223|
|(3, 3, 13, 13, 2, 2)|21.5|0.9868|
|(3, 3, 13, 13, 3, 3)|32.0|1.0519|
|(3, 3, 14, 14, 1, 1)|11.0|1.1221|
|(3, 3, 14, 14, 2, 2)|21.5|0.9863|
|(3, 3, 15, 15, 1, 1)|11.0|1.1219|
|(4, 4, 1, 1, 1, 1)|11.0|1.2021|
|(4, 4, 1, 1, 2, 2)|14.5|0.8687|
|(4, 4, 1, 1, 3, 3)|14.5|0.8248|
|(4, 4, 1, 1, 4, 4)|14.5|0.8124|
|(4, 4, 1, 1, 5, 5)|14.5|0.8066|
|(4, 4, 1, 1, 6, 6)|14.5|0.8031|
|(4, 4, 1, 1, 7, 7)|14.5|0.8009|
|(4, 4, 1, 1, 8, 8)|14.5|0.7993|
|(4, 4, 1, 1, 9, 9)|14.5|0.7981|
|(4, 4, 1, 1, 10, 10)|14.5|0.7972|
|(4, 4, 1, 1, 11, 11)|14.5|0.7965|
```

```
|(4, 4, 1, 1, 12, 12)|14.5|0.7959|
|(4, 4, 1, 1, 13, 13)|14.5|0.7954|
|(4, 4, 1, 1, 14, 14)|14.5|0.795|
|(4, 4, 2, 2, 1, 1)|11.0|1.132|
|(4, 4, 2, 2, 2, 2)|21.5|1.0338|
|(4, 4, 2, 2, 3, 3)|28.5|0.9921|
|(4, 4, 2, 2, 4, 4)|28.5|0.8639|
|(4, 4, 2, 2, 5, 5)|28.5|0.8351|
|(4, 4, 2, 2, 6, 6)|28.5|0.8224|
|(4, 4, 2, 2, 7, 7)|28.5|0.8151|
|(4, 4, 2, 2, 8, 8)|28.5|0.8104|
|(4, 4, 2, 2, 9, 9)|28.5|0.8072|
|(4, 4, 2, 2, 10, 10)|28.5|0.8048|
|(4, 4, 2, 2, 11, 11)|28.5|0.8029|
|(4, 4, 2, 2, 12, 12)|28.5|0.8014|
|(4, 4, 2, 2, 13, 13)|28.5|0.8002|
|(4, 4, 3, 3, 1, 1)|11.0|1.1214|
|(4, 4, 3, 3, 2, 2)|21.5|0.9826|
|(4, 4, 3, 3, 3, 3)|32.0|1.0431|
|(4, 4, 3, 3, 4, 4)|42.5|1.3378|
|(4, 4, 3, 3, 5, 5)|43.0|1.0521|
|(4, 4, 3, 3, 6, 6)|43.0|0.9964|
|(4, 4, 3, 3, 7, 7)|43.0|0.9739|
|(4, 4, 3, 3, 8, 8)|43.0|0.9617|
|(4, 4, 3, 3, 9, 9)|43.0|0.9541|
|(4, 4, 3, 3, 10, 10)|43.0|0.9488|
|(4, 4, 3, 3, 11, 11)|43.0|0.9449|
|(4, 4, 3, 3, 12, 12)|43.0|0.942|
|(4, 4, 4, 4, 1, 1)|11.0|1.117|
|(4, 4, 4, 4, 2, 2)|21.5|0.9689|
|(4, 4, 4, 4, 3, 3)|32.0|1.0019|
|(4, 4, 4, 4, 4, 4)|42.5|1.1074|
|(4, 4, 4, 4, 5, 5)|53.0|1.3516|
|(4, 4, 4, 4, 6, 6)|57.0|1.3272|
|(4, 4, 4, 4, 7, 7)|57.0|1.2332|
|(4, 4, 4, 4, 8, 8)|57.0|1.199|
|(4, 4, 4, 4, 9, 9)|57.0|1.1811|
|(4, 4, 4, 4, 10, 10)|57.0|1.1702|
|(4, 4, 4, 4, 11, 11)|57.0|1.1628|
|(4, 4, 5, 5, 1, 1)|11.0|1.1147|
|(4, 4, 5, 5, 2, 2)|21.5|0.9625|
|(4, 4, 5, 5, 3, 3)|32.0|0.9874|
|(4, 4, 5, 5, 4, 4)|42.5|1.0728|
|(4, 4, 5, 5, 5, 5)|53.0|1.2389|
|(4, 4, 5, 5, 6, 6)|63.5|1.6041|
|(4, 4, 5, 5, 7, 7)|71.5|2.3112|
|(4, 4, 5, 5, 8, 8)|71.5|2.1433|
|(4, 4, 5, 5, 9, 9)|71.5|2.0913|
|(4, 4, 5, 5, 10, 10)|71.5|2.0658|
|(4, 4, 6, 6, 1, 1)|11.0|1.1132|
|(4, 4, 6, 6, 2, 2)|21.5|0.9587|
|(4, 4, 6, 6, 3, 3)|32.0|0.9798|
|(4, 4, 6, 6, 4, 4)|42.5|1.0583|
|(4, 4, 6, 6, 5, 5)|53.0|1.2091|
|(4, 4, 6, 6, 6, 6)|63.5|1.5293|
|(4, 4, 6, 6, 7, 7)|72.0|2.1895|
|(4, 4, 6, 6, 8, 8)|72.0|2.0079|
|(4, 4, 6, 6, 9, 9)|72.0|1.9535|
|(4, 4, 7, 7, 1, 1)|11.0|1.1121|
|(4, 4, 7, 7, 2, 2)|21.5|0.9563|
```

```
|(4, 4, 7, 7, 3, 3)|32.0|0.9752|
|(4, 4, 7, 7, 4, 4)|42.5|1.0501|
|(4, 4, 7, 7, 5, 5)|53.0|1.1949|
|(4, 4, 7, 7, 6, 6)|63.5|1.503|
|(4, 4, 7, 7, 7, 7)|72.0|2.1419|
|(4, 4, 7, 7, 8, 8)|72.0|1.9602|
|(4, 4, 8, 8, 1, 1)|11.0|1.1114|
|(4, 4, 8, 8, 2, 2)|21.5|0.9545|
|(4, 4, 8, 8, 3, 3)|32.0|0.9721|
|(4, 4, 8, 8, 4, 4)|42.5|1.0449|
|(4, 4, 8, 8, 5, 5)|53.0|1.1865|
|(4, 4, 8, 8, 6, 6)|63.5|1.4894|
|(4, 4, 8, 8, 7, 7)|72.0|2.1208|
|(4, 4, 9, 9, 1, 1)|11.0|1.1108|
|(4, 4, 9, 9, 2, 2)|21.5|0.9532|
|(4, 4, 9, 9, 3, 3)|32.0|0.9698|
|(4, 4, 9, 9, 4, 4)|42.5|1.0413|
|(4, 4, 9, 9, 5, 5)|53.0|1.1809|
|(4, 4, 9, 9, 6, 6)|63.5|1.4809|
|(4, 4, 10, 10, 1, 1)|11.0|1.1104|
|(4, 4, 10, 10, 2, 2)|21.5|0.9522|
|(4, 4, 10, 10, 3, 3)|32.0|0.9681|
|(4, 4, 10, 10, 4, 4)|42.5|1.0386|
|(4, 4, 10, 10, 5, 5)|53.0|1.177|
|(4, 4, 11, 11, 1, 1)|11.0|1.11|
|(4, 4, 11, 11, 2, 2)|21.5|0.9514|
|(4, 4, 11, 11, 3, 3)|32.0|0.9667|
|(4, 4, 11, 11, 4, 4)|42.5|1.0366|
|(4, 4, 12, 12, 1, 1)|11.0|1.1097|
|(4, 4, 12, 12, 2, 2)|21.5|0.9508|
|(4, 4, 12, 12, 3, 3)|32.0|0.9656|
|(4, 4, 13, 13, 1, 1)|11.0|1.1094|
|(4, 4, 13, 13, 2, 2)|21.5|0.9502|
|(4, 4, 14, 14, 1, 1)|11.0|1.1092|
|(5, 5, 1, 1, 1, 1)|11.0|1.195|
|(5, 5, 1, 1, 2, 2)|14.5|0.8585|
|(5, 5, 1, 1, 3, 3)|14.5|0.8145|
|(5, 5, 1, 1, 4, 4)|14.5|0.8022|
|(5, 5, 1, 1, 5, 5)|14.5|0.7963|
|(5, 5, 1, 1, 6, 6)|14.5|0.7929|
|(5, 5, 1, 1, 7, 7)|14.5|0.7907|
|(5, 5, 1, 1, 8, 8)|14.5|0.7891|
|(5, 5, 1, 1, 9, 9)|14.5|0.7879|
|(5, 5, 1, 1, 10, 10)|14.5|0.787|
|(5, 5, 1, 1, 11, 11)|14.5|0.7863|
|(5, 5, 1, 1, 12, 12)|14.5|0.7857|
|(5, 5, 1, 1, 13, 13)|14.5|0.7852|
|(5, 5, 2, 2, 1, 1)|11.0|1.1248|
|(5, 5, 2, 2, 2, 2)|21.5|1.0155|
|(5, 5, 2, 2, 3, 3)|28.5|0.9625|
|(5, 5, 2, 2, 4, 4)|28.5|0.8343|
|(5, 5, 2, 2, 5, 5)|28.5|0.8055|
|(5, 5, 2, 2, 6, 6)|28.5|0.7927|
|(5, 5, 2, 2, 7, 7)|28.5|0.7855|
|(5, 5, 2, 2, 8, 8)|28.5|0.7808|
|(5, 5, 2, 2, 9, 9)|28.5|0.7776|
|(5, 5, 2, 2, 10, 10)|28.5|0.7752|
|(5, 5, 2, 2, 11, 11)|28.5|0.7733|
|(5, 5, 2, 2, 12, 12)|28.5|0.7718|
|(5, 5, 3, 3, 1, 1)|11.0|1.1143|
```

```
|(5, 5, 3, 3, 2, 2)|21.5|0.9643|
|(5, 5, 3, 3, 3, 3)|32.0|1.0059|
|(5, 5, 3, 3, 4, 4)|42.5|1.2659|
|(5, 5, 3, 3, 5, 5)|43.0|0.9778|
|(5, 5, 3, 3, 6, 6)|43.0|0.9221|
|(5, 5, 3, 3, 7, 7)|43.0|0.8997|
|(5, 5, 3, 3, 8, 8)|43.0|0.8875|
|(5, 5, 3, 3, 9, 9)|43.0|0.8798|
|(5, 5, 3, 3, 10, 10)|43.0|0.8745|
|(5, 5, 3, 3, 11, 11)|43.0|0.8707|
|(5, 5, 4, 4, 1, 1)|11.0|1.1099|
|(5, 5, 4, 4, 2, 2)|21.5|0.9507|
|(5, 5, 4, 4, 3, 3)|32.0|0.9648|
|(5, 5, 4, 4, 4, 4)|42.5|1.0354|
|(5, 5, 4, 4, 5, 5)|53.0|1.2071|
|(5, 5, 4, 4, 6, 6)|57.0|1.133|
|(5, 5, 4, 4, 7, 7)|57.0|1.0391|
|(5, 5, 4, 4, 8, 8)|57.0|1.0048|
|(5, 5, 4, 4, 9, 9)|57.0|0.987|
|(5, 5, 4, 4, 10, 10)|57.0|0.9761|
|(5, 5, 5, 5, 1, 1)|11.0|1.1075|
|(5, 5, 5, 5, 2, 2)|21.5|0.9443|
|(5, 5, 5, 5, 3, 3)|32.0|0.9502|
|(5, 5, 5, 5, 4, 4)|42.5|1.0009|
|(5, 5, 5, 5, 5, 5)|53.0|1.0944|
|(5, 5, 5, 5, 6, 6)|63.5|1.2683|
|(5, 5, 5, 5, 7, 7)|71.5|1.4613|
|(5, 5, 5, 5, 8, 8)|71.5|1.2934|
|(5, 5, 5, 5, 9, 9)|71.5|1.2414|
|(5, 5, 6, 6, 1, 1)|11.0|1.106|
|(5, 5, 6, 6, 2, 2)|21.5|0.9405|
|(5, 5, 6, 6, 3, 3)|32.0|0.9427|
|(5, 5, 6, 6, 4, 4)|42.5|0.9863|
|(5, 5, 6, 6, 5, 5)|53.0|1.0646|
|(5, 5, 6, 6, 6, 6)|63.5|1.1936|
|(5, 5, 6, 6, 7, 7)|74.0|1.4255|
|(5, 5, 6, 6, 8, 8)|84.5|1.97|
|(5, 5, 7, 7, 1, 1)|11.0|1.105|
|(5, 5, 7, 7, 2, 2)|21.5|0.9381|
|(5, 5, 7, 7, 3, 3)|32.0|0.9381|
|(5, 5, 7, 7, 4, 4)|42.5|0.9782|
|(5, 5, 7, 7, 5, 5)|53.0|1.0504|
|(5, 5, 7, 7, 6, 6)|63.5|1.1673|
|(5, 5, 7, 7, 7, 7)|74.0|1.3695|
|(5, 5, 8, 8, 1, 1)|11.0|1.1043|
|(5, 5, 8, 8, 2, 2)|21.5|0.9363|
|(5, 5, 8, 8, 3, 3)|32.0|0.9349|
|(5, 5, 8, 8, 4, 4)|42.5|0.973|
|(5, 5, 8, 8, 5, 5)|53.0|1.042|
|(5, 5, 8, 8, 6, 6)|63.5|1.1536|
|(5, 5, 9, 9, 1, 1)|11.0|1.1037|
|(5, 5, 9, 9, 2, 2)|21.5|0.935|
|(5, 5, 9, 9, 3, 3)|32.0|0.9327|
|(5, 5, 9, 9, 4, 4)|42.5|0.9694|
|(5, 5, 9, 9, 5, 5)|53.0|1.0364|
|(5, 5, 10, 10, 1, 1)|11.0|1.1032|
|(5, 5, 10, 10, 2, 2)|21.5|0.934|
|(5, 5, 10, 10, 3, 3)|32.0|0.9309|
|(5, 5, 10, 10, 4, 4)|42.5|0.9667|
|(5, 5, 11, 11, 1, 1)|11.0|1.1029|
```

```
|(5, 5, 11, 11, 2, 2)|21.5|0.9332|
|(5, 5, 11, 11, 3, 3)|32.0|0.9296|
|(5, 5, 12, 12, 1, 1)|11.0|1.1026|
|(5, 5, 12, 12, 2, 2)|21.5|0.9326|
|(5, 5, 13, 13, 1, 1)|11.0|1.1023|
|(6, 6, 1, 1, 1, 1)|11.0|1.1904|
|(6, 6, 1, 1, 2, 2)|14.5|0.8521|
|(6, 6, 1, 1, 3, 3)|14.5|0.8082|
|(6, 6, 1, 1, 4, 4)|14.5|0.7958|
|(6, 6, 1, 1, 5, 5)|14.5|0.7899|
|(6, 6, 1, 1, 6, 6)|14.5|0.7865|
|(6, 6, 1, 1, 7, 7)|14.5|0.7843|
|(6, 6, 1, 1, 8, 8)|14.5|0.7827|
|(6, 6, 1, 1, 9, 9)|14.5|0.7815|
|(6, 6, 1, 1, 10, 10)|14.5|0.7806|
|(6, 6, 1, 1, 11, 11)|14.5|0.7799|
|(6, 6, 1, 1, 12, 12)|14.5|0.7793|
|(6, 6, 2, 2, 1, 1)|11.0|1.1203|
|(6, 6, 2, 2, 2, 2)|21.5|1.0046|
|(6, 6, 2, 2, 3, 3)|28.5|0.9456|
|(6, 6, 2, 2, 4, 4)|28.5|0.8174|
|(6, 6, 2, 2, 5, 5)|28.5|0.7886|
|(6, 6, 2, 2, 6, 6)|28.5|0.7759|
|(6, 6, 2, 2, 7, 7)|28.5|0.7686|
|(6, 6, 2, 2, 8, 8)|28.5|0.7639|
|(6, 6, 2, 2, 9, 9)|28.5|0.7607|
|(6, 6, 2, 2, 10, 10)|28.5|0.7583|
|(6, 6, 2, 2, 11, 11)|28.5|0.7564|
|(6, 6, 3, 3, 1, 1)|11.0|1.1097|
|(6, 6, 3, 3, 2, 2)|21.5|0.9534|
|(6, 6, 3, 3, 3, 3)|32.0|0.9853|
|(6, 6, 3, 3, 4, 4)|42.5|1.23|
|(6, 6, 3, 3, 5, 5)|43.0|0.941|
|(6, 6, 3, 3, 6, 6)|43.0|0.8854|
|(6, 6, 3, 3, 7, 7)|43.0|0.8629|
|(6, 6, 3, 3, 8, 8)|43.0|0.8507|
|(6, 6, 3, 3, 9, 9)|43.0|0.8431|
|(6, 6, 3, 3, 10, 10)|43.0|0.8378|
|(6, 6, 4, 4, 1, 1)|11.0|1.1054|
|(6, 6, 4, 4, 2, 2)|21.5|0.9398|
|(6, 6, 4, 4, 3, 3)|32.0|0.9442|
|(6, 6, 4, 4, 4, 4)|42.5|0.9996|
|(6, 6, 4, 4, 5, 5)|53.0|1.1458|
|(6, 6, 4, 4, 6, 6)|57.0|1.0575|
|(6, 6, 4, 4, 7, 7)|57.0|0.9636|
|(6, 6, 4, 4, 8, 8)|57.0|0.9293|
|(6, 6, 4, 4, 9, 9)|57.0|0.9115|
|(6, 6, 5, 5, 1, 1)|11.0|1.103|
|(6, 6, 5, 5, 2, 2)|21.5|0.9334|
|(6, 6, 5, 5, 3, 3)|32.0|0.9297|
|(6, 6, 5, 5, 4, 4)|42.5|0.9651|
|(6, 6, 5, 5, 5, 5)|53.0|1.0331|
|(6, 6, 5, 5, 6, 6)|63.5|1.1608|
|(6, 6, 5, 5, 7, 7)|71.5|1.2883|
|(6, 6, 5, 5, 8, 8)|71.5|1.1204|
|(6, 6, 6, 6, 1, 1)|11.0|1.1015|
|(6, 6, 6, 6, 2, 2)|21.5|0.9296|
|(6, 6, 6, 6, 3, 3)|32.0|0.9221|
|(6, 6, 6, 6, 4, 4)|42.5|0.9505|
|(6, 6, 6, 6, 5, 5)|53.0|1.0033|
```

```
|(6, 6, 6, 6, 6, 6)|63.5|1.086|
|(6, 6, 6, 6, 7, 7)|74.0|1.2218|
|(6, 6, 7, 7, 1, 1)|11.0|1.1005|
|(6, 6, 7, 7, 2, 2)|21.5|0.9271|
|(6, 6, 7, 7, 3, 3)|32.0|0.9175|
|(6, 6, 7, 7, 4, 4)|42.5|0.9424|
|(6, 6, 7, 7, 5, 5)|53.0|0.9891|
|(6, 6, 7, 7, 6, 6)|63.5|1.0598|
|(6, 6, 8, 8, 1, 1)|11.0|1.0997|
|(6, 6, 8, 8, 2, 2)|21.5|0.9254|
|(6, 6, 8, 8, 3, 3)|32.0|0.9144|
|(6, 6, 8, 8, 4, 4)|42.5|0.9372|
|(6, 6, 8, 8, 5, 5)|53.0|0.9807|
|(6, 6, 9, 9, 1, 1)|11.0|1.0992|
|(6, 6, 9, 9, 2, 2)|21.5|0.9241|
|(6, 6, 9, 9, 3, 3)|32.0|0.9121|
|(6, 6, 9, 9, 4, 4)|42.5|0.9336|
|(6, 6, 10, 10, 1, 1)|11.0|1.0987|
|(6, 6, 10, 10, 2, 2)|21.5|0.9231|
|(6, 6, 10, 10, 3, 3)|32.0|0.9104|
|(6, 6, 11, 11, 1, 1)|11.0|1.0984|
|(6, 6, 11, 11, 2, 2)|21.5|0.9223|
|(6, 6, 12, 12, 1, 1)|11.0|1.0981|
|(7, 7, 1, 1, 1, 1)|11.0|1.1873|
|(7, 7, 1, 1, 2, 2)|14.5|0.8477|
|(7, 7, 1, 1, 3, 3)|14.5|0.8038|
|(7, 7, 1, 1, 4, 4)|14.5|0.7914|
|(7, 7, 1, 1, 5, 5)|14.5|0.7856|
|(7, 7, 1, 1, 6, 6)|14.5|0.7822|
|(7, 7, 1, 1, 7, 7)|14.5|0.7799|
|(7, 7, 1, 1, 8, 8)|14.5|0.7783|
|(7, 7, 1, 1, 9, 9)|14.5|0.7772|
|(7, 7, 1, 1, 10, 10)|14.5|0.7763|
|(7, 7, 1, 1, 11, 11)|14.5|0.7755|
|(7, 7, 2, 2, 1, 1)|11.0|1.1172|
|(7, 7, 2, 2, 2, 2)|21.5|0.9974|
|(7, 7, 2, 2, 3, 3)|28.5|0.9347|
|(7, 7, 2, 2, 4, 4)|28.5|0.8065|
|(7, 7, 2, 2, 5, 5)|28.5|0.7777|
|(7, 7, 2, 2, 6, 6)|28.5|0.7649|
|(7, 7, 2, 2, 7, 7)|28.5|0.7577|
|(7, 7, 2, 2, 8, 8)|28.5|0.753|
|(7, 7, 2, 2, 9, 9)|28.5|0.7498|
|(7, 7, 2, 2, 10, 10)|28.5|0.7473|
|(7, 7, 3, 3, 1, 1)|11.0|1.1066|
|(7, 7, 3, 3, 2, 2)|21.5|0.9462|
|(7, 7, 3, 3, 3, 3)|32.0|0.9723|
|(7, 7, 3, 3, 4, 4)|42.5|1.2085|
|(7, 7, 3, 3, 5, 5)|43.0|0.9191|
|(7, 7, 3, 3, 6, 6)|43.0|0.8634|
|(7, 7, 3, 3, 7, 7)|43.0|0.8409|
|(7, 7, 3, 3, 8, 8)|43.0|0.8288|
|(7, 7, 3, 3, 9, 9)|43.0|0.8211|
|(7, 7, 4, 4, 1, 1)|11.0|1.1023|
|(7, 7, 4, 4, 2, 2)|21.5|0.9325|
|(7, 7, 4, 4, 3, 3)|32.0|0.9311|
|(7, 7, 4, 4, 4, 4)|42.5|0.9781|
|(7, 7, 4, 4, 5, 5)|53.0|1.1118|
|(7, 7, 4, 4, 6, 6)|57.0|1.0172|
|(7, 7, 4, 4, 7, 7)|57.0|0.9232|
```

```
|(7, 7, 4, 4, 8, 8)|57.0|0.889|
|(7, 7, 5, 5, 1, 1)|11.0|1.0999|
|(7, 7, 5, 5, 2, 2)|21.5|0.9261|
|(7, 7, 5, 5, 3, 3)|32.0|0.9166|
|(7, 7, 5, 5, 4, 4)|42.5|0.9436|
|(7, 7, 5, 5, 5, 5)|53.0|0.9992|
|(7, 7, 5, 5, 6, 6)|63.5|1.1074|
|(7, 7, 5, 5, 7, 7)|71.5|1.212|
|(7, 7, 6, 6, 1, 1)|11.0|1.0984|
|(7, 7, 6, 6, 2, 2)|21.5|0.9223|
|(7, 7, 6, 6, 3, 3)|32.0|0.909|
|(7, 7, 6, 6, 4, 4)|42.5|0.929|
|(7, 7, 6, 6, 5, 5)|53.0|0.9693|
|(7, 7, 6, 6, 6, 6)|63.5|1.0326|
|(7, 7, 7, 7, 1, 1)|11.0|1.0974|
|(7, 7, 7, 7, 2, 2)|21.5|0.9199|
|(7, 7, 7, 7, 3, 3)|32.0|0.9044|
|(7, 7, 7, 7, 4, 4)|42.5|0.9209|
|(7, 7, 7, 7, 5, 5)|53.0|0.9551|
|(7, 7, 8, 8, 1, 1)|11.0|1.0966|
|(7, 7, 8, 8, 2, 2)|21.5|0.9181|
|(7, 7, 8, 8, 3, 3)|32.0|0.9013|
|(7, 7, 8, 8, 4, 4)|42.5|0.9157|
|(7, 7, 9, 9, 1, 1)|11.0|1.096|
|(7, 7, 9, 9, 2, 2)|21.5|0.9168|
|(7, 7, 9, 9, 3, 3)|32.0|0.899|
|(7, 7, 10, 10, 1, 1)|11.0|1.0956|
|(7, 7, 10, 10, 2, 2)|21.5|0.9158|
|(7, 7, 11, 11, 1, 1)|11.0|1.0952|
|(8, 8, 1, 1, 1, 1)|11.0|1.185|
|(8, 8, 1, 1, 2, 2)|14.5|0.8446|
|(8, 8, 1, 1, 3, 3)|14.5|0.8006|
|(8, 8, 1, 1, 4, 4)|14.5|0.7883|
|(8, 8, 1, 1, 5, 5)|14.5|0.7824|
|(8, 8, 1, 1, 6, 6)|14.5|0.779|
|(8, 8, 1, 1, 7, 7)|14.5|0.7768|
|(8, 8, 1, 1, 8, 8)|14.5|0.7752|
|(8, 8, 1, 1, 9, 9)|14.5|0.774|
|(8, 8, 1, 1, 10, 10)|14.5|0.7731|
|(8, 8, 2, 2, 1, 1)|11.0|1.1149|
|(8, 8, 2, 2, 2, 2)|21.5|0.9922|
|(8, 8, 2, 2, 3, 3)|28.5|0.9271|
|(8, 8, 2, 2, 4, 4)|28.5|0.7988|
|(8, 8, 2, 2, 5, 5)|28.5|0.7701|
|(8, 8, 2, 2, 6, 6)|28.5|0.7573|
|(8, 8, 2, 2, 7, 7)|28.5|0.7501|
|(8, 8, 2, 2, 8, 8)|28.5|0.7454|
|(8, 8, 2, 2, 9, 9)|28.5|0.7421|
|(8, 8, 3, 3, 1, 1)|11.0|1.1043|
|(8, 8, 3, 3, 2, 2)|21.5|0.941|
|(8, 8, 3, 3, 3, 3)|32.0|0.9632|
|(8, 8, 3, 3, 4, 4)|42.5|1.1942|
|(8, 8, 3, 3, 5, 5)|43.0|0.9045|
|(8, 8, 3, 3, 6, 6)|43.0|0.8488|
|(8, 8, 3, 3, 7, 7)|43.0|0.8263|
|(8, 8, 3, 3, 8, 8)|43.0|0.8142|
|(8, 8, 4, 4, 1, 1)|11.0|1.1|
|(8, 8, 4, 4, 2, 2)|21.5|0.9273|
|(8, 8, 4, 4, 3, 3)|32.0|0.9221|
|(8, 8, 4, 4, 4, 4)|42.5|0.9638|
```

```
|(8, 8, 4, 4, 5, 5)|53.0|1.0902|
|(8, 8, 4, 4, 6, 6)|57.0|0.9921|
|(8, 8, 4, 4, 7, 7)|57.0|0.8981|
|(8, 8, 5, 5, 1, 1)|11.0|1.0976|
|(8, 8, 5, 5, 2, 2)|21.5|0.9209|
|(8, 8, 5, 5, 3, 3)|32.0|0.9075|
|(8, 8, 5, 5, 4, 4)|42.5|0.9293|
|(8, 8, 5, 5, 5, 5)|53.0|0.9776|
|(8, 8, 5, 5, 6, 6)|63.5|1.0754|
|(8, 8, 6, 6, 1, 1)|11.0|1.0961|
|(8, 8, 6, 6, 2, 2)|21.5|0.9171|
|(8, 8, 6, 6, 3, 3)|32.0|0.9|
|(8, 8, 6, 6, 4, 4)|42.5|0.9147|
|(8, 8, 6, 6, 5, 5)|53.0|0.9477|
|(8, 8, 7, 7, 1, 1)|11.0|1.0951|
|(8, 8, 7, 7, 2, 2)|21.5|0.9147|
|(8, 8, 7, 7, 3, 3)|32.0|0.8953|
|(8, 8, 7, 7, 4, 4)|42.5|0.9066|
|(8, 8, 8, 8, 1, 1)|11.0|1.0943|
|(8, 8, 8, 8, 2, 2)|21.5|0.9129|
|(8, 8, 8, 8, 3, 3)|32.0|0.8922|
|(8, 8, 9, 9, 1, 1)|11.0|1.0938|
|(8, 8, 9, 9, 2, 2)|21.5|0.9116|
|(8, 8, 10, 10, 1, 1)|11.0|1.0933|
|(9, 9, 1, 1, 1, 1)|11.0|1.1833|
|(9, 9, 1, 1, 2, 2)|14.5|0.8422|
|(9, 9, 1, 1, 3, 3)|14.5|0.7982|
|(9, 9, 1, 1, 4, 4)|14.5|0.7859|
|(9, 9, 1, 1, 5, 5)|14.5|0.78|
|(9, 9, 1, 1, 6, 6)|14.5|0.7766|
|(9, 9, 1, 1, 7, 7)|14.5|0.7744|
|(9, 9, 1, 1, 8, 8)|14.5|0.7728|
|(9, 9, 1, 1, 9, 9)|14.5|0.7716|
|(9, 9, 2, 2, 1, 1)|11.0|1.1131|
|(9, 9, 2, 2, 2, 2)|21.5|0.9883|
|(9, 9, 2, 2, 3, 3)|28.5|0.9214|
|(9, 9, 2, 2, 4, 4)|28.5|0.7932|
|(9, 9, 2, 2, 5, 5)|28.5|0.7644|
|(9, 9, 2, 2, 6, 6)|28.5|0.7517|
|(9, 9, 2, 2, 7, 7)|28.5|0.7444|
|(9, 9, 2, 2, 8, 8)|28.5|0.7397|
|(9, 9, 3, 3, 1, 1)|11.0|1.1026|
|(9, 9, 3, 3, 2, 2)|21.5|0.9371|
|(9, 9, 3, 3, 3, 3)|32.0|0.9566|
|(9, 9, 3, 3, 4, 4)|42.5|1.184|
|(9, 9, 3, 3, 5, 5)|43.0|0.8941|
|(9, 9, 3, 3, 6, 6)|43.0|0.8384|
|(9, 9, 3, 3, 7, 7)|43.0|0.8159|
|(9, 9, 4, 4, 1, 1)|11.0|1.0982|
|(9, 9, 4, 4, 2, 2)|21.5|0.9234|
|(9, 9, 4, 4, 3, 3)|32.0|0.9154|
|(9, 9, 4, 4, 4, 4)|42.5|0.9536|
|(9, 9, 4, 4, 5, 5)|53.0|1.0753|
|(9, 9, 4, 4, 6, 6)|57.0|0.975|
|(9, 9, 5, 5, 1, 1)|11.0|1.0959|
|(9, 9, 5, 5, 2, 2)|21.5|0.917|
|(9, 9, 5, 5, 3, 3)|32.0|0.9009|
|(9, 9, 5, 5, 4, 4)|42.5|0.9191|
|(9, 9, 5, 5, 5, 5)|53.0|0.9626|
|(9, 9, 6, 6, 1, 1)|11.0|1.0944|
```

```
|(9, 9, 6, 6, 2, 2)|21.5|0.9133|
|(9, 9, 6, 6, 3, 3)|32.0|0.8933|
|(9, 9, 6, 6, 4, 4)|42.5|0.9045|
|(9, 9, 7, 7, 1, 1)|11.0|1.0933|
|(9, 9, 7, 7, 2, 2)|21.5|0.9108|
|(9, 9, 7, 7, 3, 3)|32.0|0.8887|
|(9, 9, 8, 8, 1, 1)|11.0|1.0926|
|(9, 9, 8, 8, 2, 2)|21.5|0.909|
|(9, 9, 9, 9, 1, 1)|11.0|1.092|
|(10, 10, 1, 1, 1, 1)|11.0|1.1819|
|(10, 10, 1, 1, 2, 2)|14.5|0.8403|
|(10, 10, 1, 1, 3, 3)|14.5|0.7963|
|(10, 10, 1, 1, 4, 4)|14.5|0.784|
|(10, 10, 1, 1, 5, 5)|14.5|0.7781|
|(10, 10, 1, 1, 6, 6)|14.5|0.7747|
|(10, 10, 1, 1, 7, 7)|14.5|0.7725|
|(10, 10, 1, 1, 8, 8)|14.5|0.7709|
|(10, 10, 2, 2, 1, 1)|11.0|1.1118|
|(10, 10, 2, 2, 2, 2)|21.5|0.9853|
|(10, 10, 2, 2, 3, 3)|28.5|0.9171|
|(10, 10, 2, 2, 4, 4)|28.5|0.7889|
|(10, 10, 2, 2, 5, 5)|28.5|0.7601|
|(10, 10, 2, 2, 6, 6)|28.5|0.7473|
|(10, 10, 2, 2, 7, 7)|28.5|0.7401|
|(10, 10, 3, 3, 1, 1)|11.0|1.1012|
|(10, 10, 3, 3, 2, 2)|21.5|0.9341|
|(10, 10, 3, 3, 3, 3)|32.0|0.9515|
|(10, 10, 3, 3, 4, 4)|42.5|1.1764|
|(10, 10, 3, 3, 5, 5)|43.0|0.8863|
|(10, 10, 3, 3, 6, 6)|43.0|0.8306|
|(10, 10, 4, 4, 1, 1)|11.0|1.0968|
|(10, 10, 4, 4, 2, 2)|21.5|0.9204|
|(10, 10, 4, 4, 3, 3)|32.0|0.9103|
|(10, 10, 4, 4, 4, 4)|42.5|0.9459|
|(10, 10, 4, 4, 5, 5)|53.0|1.0643|
|(10, 10, 5, 5, 1, 1)|11.0|1.0945|
|(10, 10, 5, 5, 2, 2)|21.5|0.914|
|(10, 10, 5, 5, 3, 3)|32.0|0.8958|
|(10, 10, 5, 5, 4, 4)|42.5|0.9114|
|(10, 10, 6, 6, 1, 1)|11.0|1.093|
|(10, 10, 6, 6, 2, 2)|21.5|0.9102|
|(10, 10, 6, 6, 3, 3)|32.0|0.8883|
|(10, 10, 7, 7, 1, 1)|11.0|1.0919|
|(10, 10, 7, 7, 2, 2)|21.5|0.9078|
|(10, 10, 8, 8, 1, 1)|11.0|1.0912|
|(11, 11, 1, 1, 1, 1)|11.0|1.1808|
|(11, 11, 1, 1, 2, 2)|14.5|0.8387|
|(11, 11, 1, 1, 3, 3)|14.5|0.7948|
|(11, 11, 1, 1, 4, 4)|14.5|0.7825|
|(11, 11, 1, 1, 5, 5)|14.5|0.7766|
|(11, 11, 1, 1, 6, 6)|14.5|0.7732|
|(11, 11, 1, 1, 7, 7)|14.5|0.7709|
|(11, 11, 2, 2, 1, 1)|11.0|1.1106|
|(11, 11, 2, 2, 2, 2)|21.5|0.9828|
|(11, 11, 2, 2, 3, 3)|28.5|0.9136|
|(11, 11, 2, 2, 4, 4)|28.5|0.7854|
|(11, 11, 2, 2, 5, 5)|28.5|0.7567|
|(11, 11, 2, 2, 6, 6)|28.5|0.7439|
|(11, 11, 3, 3, 1, 1)|11.0|1.1001|
|(11, 11, 3, 3, 2, 2)|21.5|0.9316|
```

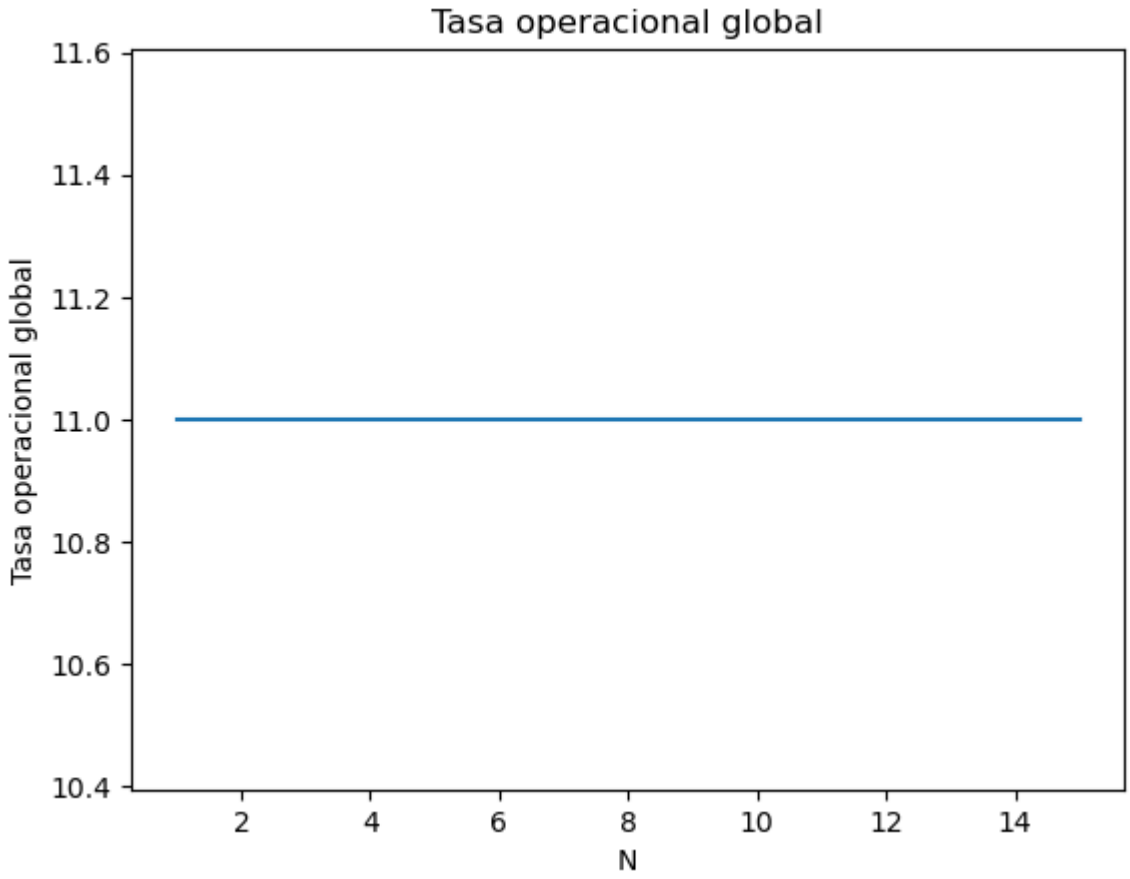
```
|(11, 11, 3, 3, 3, 3)|32.0|0.9475|
|(11, 11, 3, 3, 4, 4)|42.5|1.1704|
|(11, 11, 3, 3, 5, 5)|43.0|0.8802|
|(11, 11, 4, 4, 1, 1)|11.0|1.0957|
|(11, 11, 4, 4, 2, 2)|21.5|0.918|
|(11, 11, 4, 4, 3, 3)|32.0|0.9063|
|(11, 11, 4, 4, 4, 4)|42.5|0.94|
|(11, 11, 5, 5, 1, 1)|11.0|1.0934|
|(11, 11, 5, 5, 2, 2)|21.5|0.9116|
|(11, 11, 5, 5, 3, 3)|32.0|0.8918|
|(11, 11, 6, 6, 1, 1)|11.0|1.0919|
|(11, 11, 6, 6, 2, 2)|21.5|0.9078|
|(11, 11, 7, 7, 1, 1)|11.0|1.0908|
|(12, 12, 1, 1, 1, 1)|11.0|1.1798|
|(12, 12, 1, 1, 2, 2)|14.5|0.8375|
|(12, 12, 1, 1, 3, 3)|14.5|0.7936|
|(12, 12, 1, 1, 4, 4)|14.5|0.7812|
|(12, 12, 1, 1, 5, 5)|14.5|0.7753|
|(12, 12, 1, 1, 6, 6)|14.5|0.7719|
|(12, 12, 2, 2, 1, 1)|11.0|1.1097|
|(12, 12, 2, 2, 2, 2)|21.5|0.9809|
|(12, 12, 2, 2, 3, 3)|28.5|0.9108|
|(12, 12, 2, 2, 4, 4)|28.5|0.7826|
|(12, 12, 2, 2, 5, 5)|28.5|0.7539|
|(12, 12, 3, 3, 1, 1)|11.0|1.0991|
|(12, 12, 3, 3, 2, 2)|21.5|0.9297|
|(12, 12, 3, 3, 3, 3)|32.0|0.9442|
|(12, 12, 3, 3, 4, 4)|42.5|1.1656|
|(12, 12, 4, 4, 1, 1)|11.0|1.0948|
|(12, 12, 4, 4, 2, 2)|21.5|0.916|
|(12, 12, 4, 4, 3, 3)|32.0|0.9031|
|(12, 12, 5, 5, 1, 1)|11.0|1.0924|
|(12, 12, 5, 5, 2, 2)|21.5|0.9096|
|(12, 12, 6, 6, 1, 1)|11.0|1.0909|
|(13, 13, 1, 1, 1, 1)|11.0|1.1791|
|(13, 13, 1, 1, 2, 2)|14.5|0.8364|
|(13, 13, 1, 1, 3, 3)|14.5|0.7925|
|(13, 13, 1, 1, 4, 4)|14.5|0.7801|
|(13, 13, 1, 1, 5, 5)|14.5|0.7743|
|(13, 13, 2, 2, 1, 1)|11.0|1.1089|
|(13, 13, 2, 2, 2, 2)|21.5|0.9792|
|(13, 13, 2, 2, 3, 3)|28.5|0.9085|
|(13, 13, 2, 2, 4, 4)|28.5|0.7803|
|(13, 13, 3, 3, 1, 1)|11.0|1.0984|
|(13, 13, 3, 3, 2, 2)|21.5|0.928|
|(13, 13, 3, 3, 3, 3)|32.0|0.9415|
|(13, 13, 4, 4, 1, 1)|11.0|1.094|
|(13, 13, 4, 4, 2, 2)|21.5|0.9144|
|(13, 13, 5, 5, 1, 1)|11.0|1.0917|
|(14, 14, 1, 1, 1, 1)|11.0|1.1784|
|(14, 14, 1, 1, 2, 2)|14.5|0.8355|
|(14, 14, 1, 1, 3, 3)|14.5|0.7916|
|(14, 14, 1, 1, 4, 4)|14.5|0.7793|
|(14, 14, 2, 2, 1, 1)|11.0|1.1083|
|(14, 14, 2, 2, 2, 2)|21.5|0.9778|
|(14, 14, 2, 2, 3, 3)|28.5|0.9066|
|(14, 14, 3, 3, 1, 1)|11.0|1.0977|
|(14, 14, 3, 3, 2, 2)|21.5|0.9266|
|(14, 14, 4, 4, 1, 1)|11.0|1.0934|
|(15, 15, 1, 1, 1, 1)|11.0|1.1778|
```

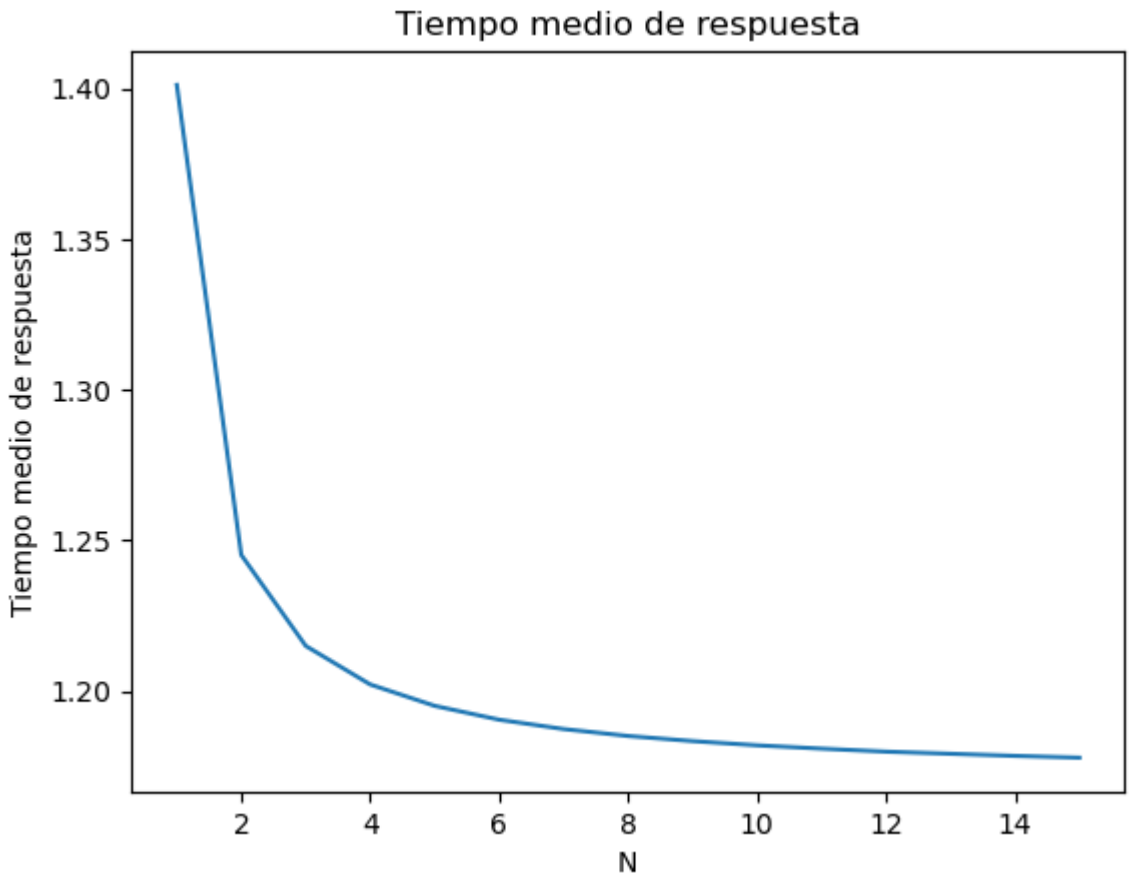
```
|(15, 15, 1, 1, 2, 2)|14.5|0.8348|
|(15, 15, 1, 1, 3, 3)|14.5|0.7908|
|(15, 15, 2, 2, 1, 1)|11.0|1.1077|
|(15, 15, 2, 2, 2, 2)|21.5|0.9766|
|(15, 15, 3, 3, 1, 1)|11.0|1.0971|
|(16, 16, 1, 1, 1, 1)|11.0|1.1773|
|(16, 16, 1, 1, 2, 2)|14.5|0.8341|
|(16, 16, 2, 2, 1, 1)|11.0|1.1072|
```

Cambios en el nº de servidores de web

In [13]:
```
# Un grafico de tasa global y otro de tiempo medio de respuesta desde base de da
# Filtrar las combinaciones de N tal que N sea de la forma (x,x,1,1,1,1)
combinaciones = [
    x
    for x in base_datos_aprendizaje
    if x[0] in [(i, i, 1, 1, 1, 1) for i in range(1, 16)]
]

print(combinaciones)

# Grafico de tasa global
plt.plot([x[0][0] for x in combinaciones], [x[1] for x in combinaciones])
plt.title("Tasa operacional global")
plt.xlabel("N")
plt.ylabel("Tasa operacional global")
plt.show()

# Grafico de tiempo medio de respuesta
plt.plot([x[0][0] for x in combinaciones], [x[2] for x in combinaciones])
plt.title("Tiempo medio de respuesta")
plt.xlabel("N")
plt.ylabel("Tiempo medio de respuesta")
plt.show()
```

```
[((1, 1, 1, 1, 1, 1), 11.0, 1.4011), ((2, 2, 1, 1, 1, 1), 11.0, 1.2451), ((3, 3,
1, 1, 1, 1), 11.0, 1.2149), ((4, 4, 1, 1, 1, 1), 11.0, 1.2021), ((5, 5, 1, 1, 1,
1), 11.0, 1.195), ((6, 6, 1, 1, 1, 1), 11.0, 1.1904), ((7, 7, 1, 1, 1, 1), 11.0,
1.1873), ((8, 8, 1, 1, 1, 1), 11.0, 1.185), ((9, 9, 1, 1, 1, 1), 11.0, 1.1833),
((10, 10, 1, 1, 1, 1), 11.0, 1.1819), ((11, 11, 1, 1, 1, 1), 11.0, 1.1808), ((12,
12, 1, 1, 1, 1), 11.0, 1.1798), ((13, 13, 1, 1, 1, 1), 11.0, 1.1791), ((14, 14,
1, 1, 1, 1), 11.0, 1.1784), ((15, 15, 1, 1, 1, 1), 11.0, 1.1778)]
```



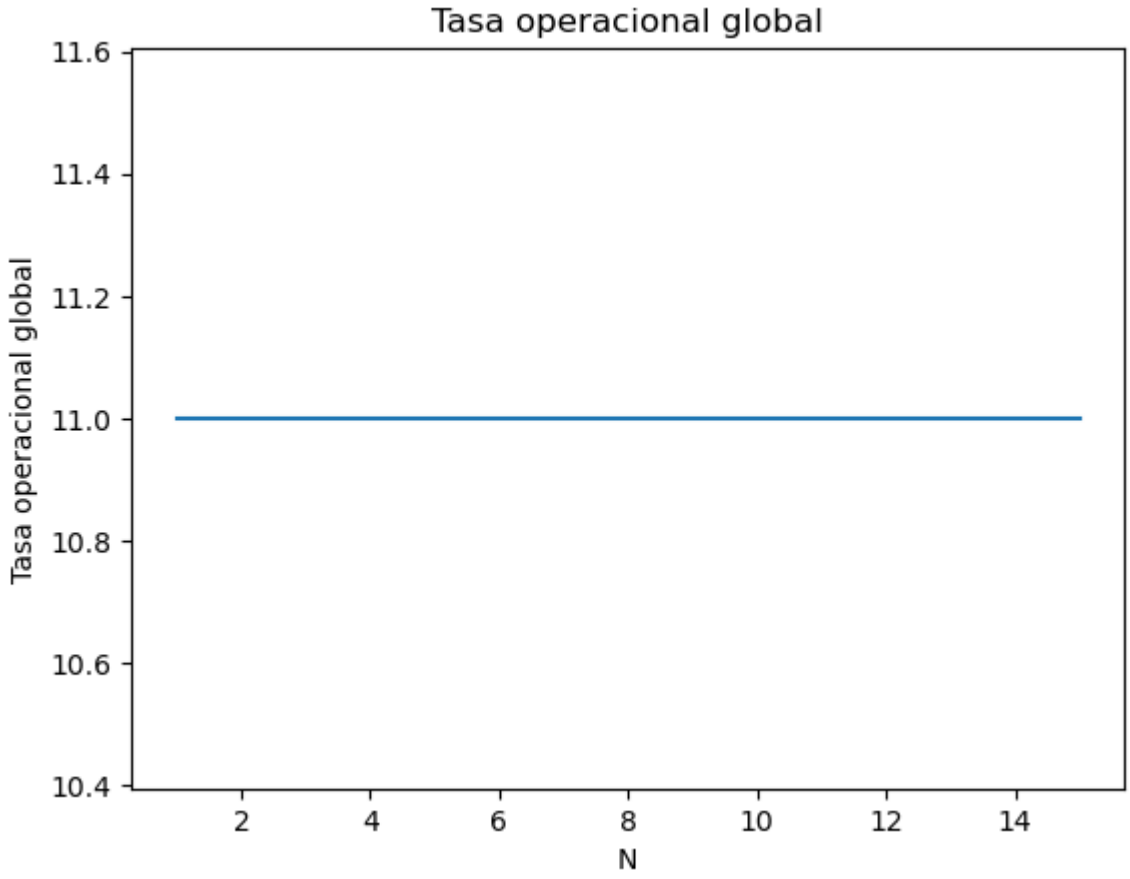


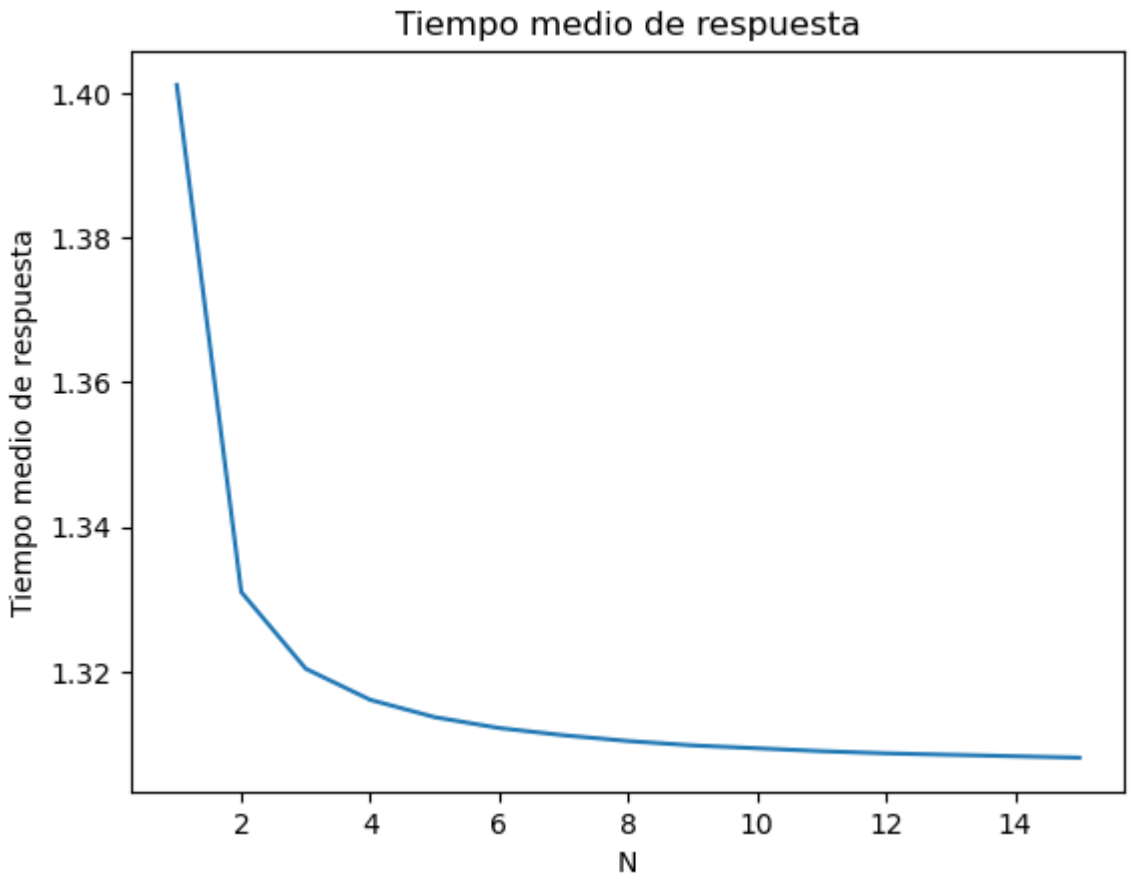
Cambios en el nº de servidores de aplicación

In [14]:
```
# Un grafico de tasa global y otro de tiempo medio de respuesta desde base de da
# Filtrar las combinaciones de N tal que N sea de la forma (1,1,x,x,1,1)
```

```
combinaciones = [
    x
    for x in base_datos_aprendizaje
    if x[0] in [(1, 1, i, i, 1, 1) for i in range(1, 16)]
]

print(combinaciones)

# Grafico de tasa global
plt.plot([x[0][2] for x in combinaciones], [x[1] for x in combinaciones])
plt.title("Tasa operacional global")
plt.xlabel("N")
plt.ylabel("Tasa operacional global")
plt.show()

# Grafico de tiempo medio de respuesta
plt.plot([x[0][2] for x in combinaciones], [x[2] for x in combinaciones])
plt.title("Tiempo medio de respuesta")
plt.xlabel("N")
plt.ylabel("Tiempo medio de respuesta")
plt.show()
```

```
[((1, 1, 1, 1, 1, 1), 11.0, 1.4011), ((1, 1, 2, 2, 1, 1), 11.0, 1.331), ((1, 1,
3, 3, 1, 1), 11.0, 1.3204), ((1, 1, 4, 4, 1, 1), 11.0, 1.3161), ((1, 1, 5, 5, 1,
1), 11.0, 1.3137), ((1, 1, 6, 6, 1, 1), 11.0, 1.3122), ((1, 1, 7, 7, 1, 1), 11.0,
1.3112), ((1, 1, 8, 8, 1, 1), 11.0, 1.3104), ((1, 1, 9, 9, 1, 1), 11.0, 1.3098),
((1, 1, 10, 10, 1, 1), 11.0, 1.3094), ((1, 1, 11, 11, 1, 1), 11.0, 1.309), ((1,
1, 12, 12, 1, 1), 11.0, 1.3087), ((1, 1, 13, 13, 1, 1), 11.0, 1.3085), ((1, 1, 1
4, 14, 1, 1), 11.0, 1.3083), ((1, 1, 15, 15, 1, 1), 11.0, 1.3081)]
```

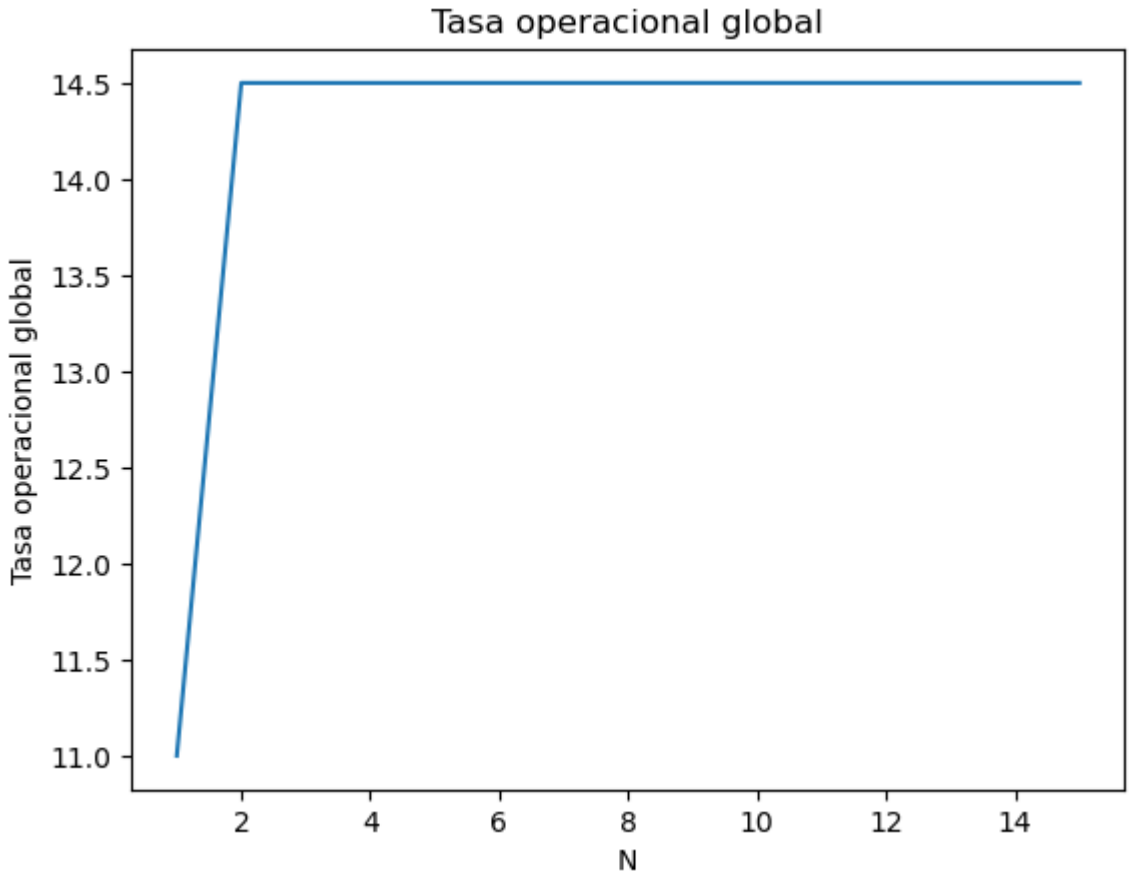


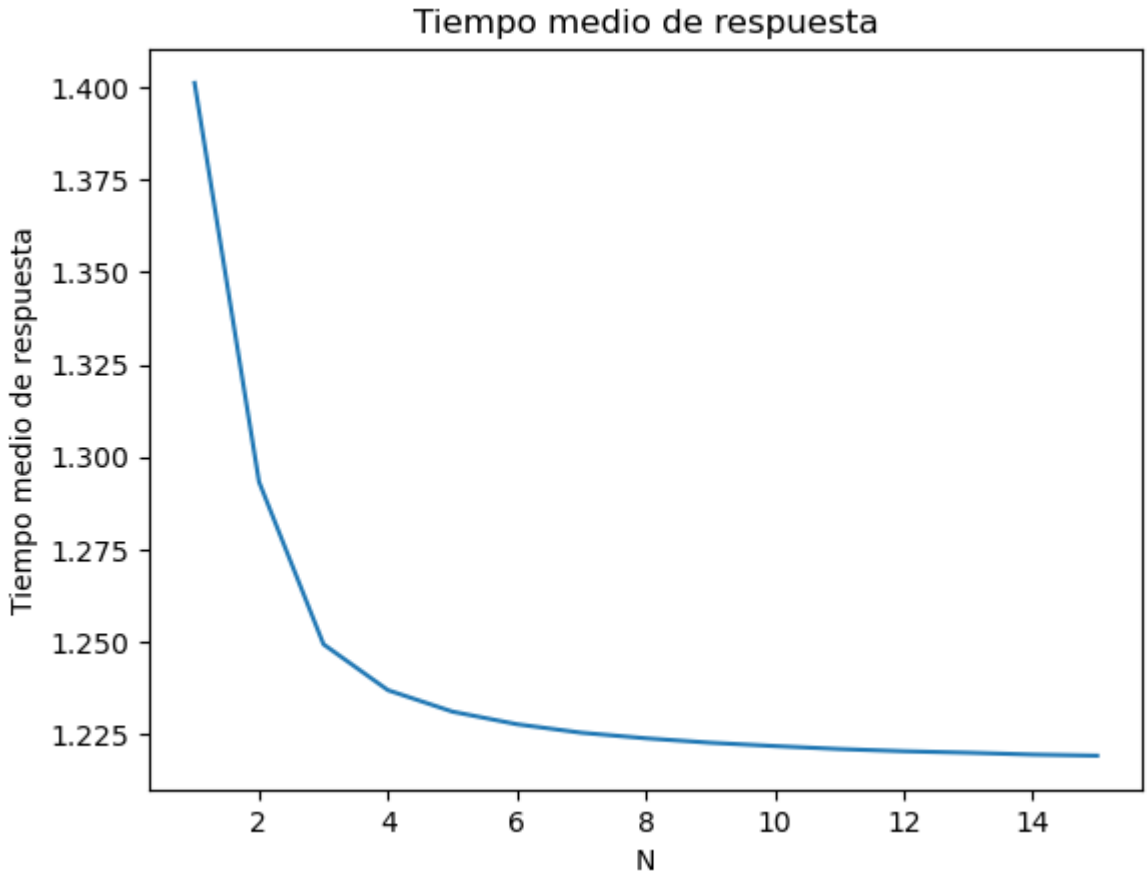


Cambios en el nº de servidores de base de datos

In [15]:
```
# Un grafico de tasa global y otro de tiempo medio de respuesta desde base de da
# Filtrar las combinaciones de N tal que N sea de la forma (1,1,1,1,x,x)
combinaciones = [
    x
    for x in base_datos_aprendizaje
    if x[0] in [(1, 1, 1, 1, i, i) for i in range(1, 16)]
]

print(combinaciones)

# Grafico de tasa global
plt.plot([x[0][4] for x in combinaciones], [x[1] for x in combinaciones])
plt.title("Tasa operacional global")
plt.xlabel("N")
plt.ylabel("Tasa operacional global")
plt.show()

# Grafico de tiempo medio de respuesta
plt.plot([x[0][4] for x in combinaciones], [x[2] for x in combinaciones])
plt.title("Tiempo medio de respuesta")
plt.xlabel("N")
plt.ylabel("Tiempo medio de respuesta")
plt.show()
```

```
[((1, 1, 1, 1, 1, 1), 11.0, 1.4011), ((1, 1, 1, 1, 2, 2), 14.5, 1.2933), ((1, 1,
1, 1, 3, 3), 14.5, 1.2494), ((1, 1, 1, 1, 4, 4), 14.5, 1.237), ((1, 1, 1, 1, 5,
5), 14.5, 1.2312), ((1, 1, 1, 1, 6, 6), 14.5, 1.2278), ((1, 1, 1, 1, 7, 7), 14.5,
1.2255), ((1, 1, 1, 1, 8, 8), 14.5, 1.224), ((1, 1, 1, 1, 9, 9), 14.5, 1.2228),
((1, 1, 1, 1, 10, 10), 14.5, 1.2219), ((1, 1, 1, 1, 11, 11), 14.5, 1.2211), ((1,
1, 1, 1, 12, 12), 14.5, 1.2205), ((1, 1, 1, 1, 13, 13), 14.5, 1.2201), ((1, 1, 1,
1, 14, 14), 14.5, 1.2196), ((1, 1, 1, 1, 15, 15), 14.5, 1.2193)]
```




Tiempo medio de respuesta
Tiempo medio de respuesta
1.400
1.375
1.350
1.325
1.300
1.275
1.250
1.225
2
4
6
8
10
12
14
N